Journal of Software

ISSN 1796-217X

Volume 6, Number 12, December 2011

Contents

Special Issue: Parallel and Distributed Data Processing

Guest Editor: Xianchao Zhang and Feng Xia

REGULAR PAPERS

Special Issue on Parallel and Distributed Data Processing

Guest Editorial

As an exponentially increasing amount of data is generated everyday, parallel and distributed data processing has emerged as a key enabling technology and plays more and more important roles in modern computing and information technologies. In recent years, many researchers have become increasingly interested in the field of parallel and distributed data processing, and a large number of remarkable academic achievements have been published and applied to various domains.

In this special issue, we selected some excellent papers from the third International Symposium on Parallel Architectures, Algorithms and Programming (PAAP 2010), which was held in Dalian, China, December 18-20, 2010. In addition, we invited and selected some representative research papers in the broad area of parallel and distributed data processing.

The paper titled "A Translation Framework for Executing the Sequential Binary Code on CPU/GPU Based Architectures" proposes a novel translation framework for constructing the virtual execution environment aiming at accelerating the process of DBT on CPU/GPU based architectures. With parallelizable parts (hot spots) of binary code and their related information, the framework converts the sequential code into PTX form and executes them on GPUs.

The paper titled "A Dynamic-Static Combined Code Layout Reorganization Approach for Dynamic Binary Translation" presents a dynamic-static combined approach to reorganize the layout of software cache. Under this approach, we first employ an emulating execution to collect the profile information and the translated target code.

The paper titled "An Efficient Hybrid Clustering-PSO Algorithm for Anomaly Intrusion Detection" demonstrates IDCPSO algorithm, which combines an unsupervised clustering algorithm with the PSO technique to optimize the clustering results and obtain the optimal detection result with unlabeled data.

The paper titled "A Hybrid Method for XML Clustering by Structure and Content" presents an effective XML cluster method based both structural similarity calculated using method PFWLCS with position frequency weight, based on frequency path model, and content information contained in XML files.

The paper titled "Hybrid Distributed Shared Memory Space in Multi-core Processors" observes that it is unnecessary to perform V2P address translation for private data accesses and introduces hybrid DSM organization and run-time partitioning technique in order to improve the system performance by reducing V2P address translation overhead as much as possible.

The paper titled "Design and Evaluation of an Online Anomaly Detector for Distributed Storage Systems" exploits the stable relationship between workloads and system resource statistics to detect the performance anomaly and identify faulty sources which cause the performance anomaly in the system.

The paper titled "Multi-pattern matching with wildcards" presents efficient algorithms of multi-pattern matching with wildcards based on the fast Fourier transform and demonstrates an FFT implementation based on the modular arithmetic for machines with 64-bit word.

The paper titled "RPPA: A Remote Parallel Program Performance Analysis Tool" presents a remote parallel program performance analysis tool, RPPA (Remote Parallel Performance Analyzer), which is based on dynamic code instrumentation.

The paper titled "Topic Detection with Hypergraph Partition Algorithm" proposes SMHP (Similarity Matrix based Hypergraph Partition) algorithm, which aims at improving the efficiency of Topic Detection.

The paper titled "Query by Humming Systems Using Melody Matching Model Based on the Genetic Algorithm" discusses a system called Query by humming (QBH). QBH refers to music information retrieval systems where short audio clips of singing or humming act as queries. This paper proposes QBH using melody matching model based on the genetic algorithm and improving the ranking result by local sensitive hashing algorithm.

The paper titled "Segmenting Webpage with Gomory-Hu Tree Based Clustering" proposes a novel web page segmentation algorithm based on finding the Gomory-Hu tree in a planar graph. The algorithm firstly distills vision and structure information from a web page to construct a weighted undirected graph, whose vertices are the leaf nodes of the DOM tree and the edges represent the visible position relationship between vertices. Then it partitions the graph with Gomory-Hu tree based clustering algorithm.

The paper titled "A Two-Dimension XML Encoding Method based on Variable Length Binary Code" considers the issue of the insertion operation of intermediate nodes into XML documents using update-friendly XML labeling schemes, and proposes an efficient two-dimension XML encoding method based on the variable length binary code, DO-VLEI code.

The paper titled “A Hop-by-hop Cross-layer Congestion Control Scheme for Wireless Sensor Networks” addresses congestions in wireless sensor networks, and presents a hop-by-hop cross-layer congestion control scheme built on contention-based MAC protocol. Simulations have been conducted to compare the proposed scheme against closely-related existing schemes.

The paper titled “Role Assorted Community Discovery for Weighted Networks” considers the difficulties in

doi:10.4304/jsw.6.12.2329-2330

community discovery. The authors show that this problem can be solved according to the role assorted method which gives distinguish labels to vertices in the same community. The method leads to a number of possible algorithms for detecting community structures in both unweighted and weighted networks.

The paper titled "A Survey on Particle Swarm Optimization Algorithms for Multimodal Function Optimization" gives a comprehensive review of existing works done in the field of multimodal function optimization as well as a critical analysis of the existing methods.

It has been a great pleasure to run this special issue, which reveals important research results in the field of parallel and distributed data processing. We would like to thank Prof. Kassem Saleh, Editor-in-Chief of Journal of Software, and Dr. George J. Sun, Executive Editor of Academy Publisher, for giving us the opportunity to organize this special issue and for their great help in the organization of this issue. We thank all authors for their submissions and all reviewers for their diligent work in evaluating these submissions. We sincerely hope that you enjoy reading these distinguished papers.

Guest Editors

Xianchao Zhang
School of Software, Dalian University of Technology, China
Email: xczhang@dlut.edu.cn

Feng Xia
School of Software, Dalian University of Technology, China
Email: f.xia@ieee.org

Xianchao Zhang is a Full Professor at Dalian University of Technology, China. He received his B.S degree in Applied Mathematics and M.S. degree in Computational Mathematics from National university of Defense Technology in 1994 and 1998, respectively. He received his Ph.D. in Computer Theory and Software from University of Science and Technology of China in 2000. He joined Dalian University of Technology in 2003 after 2 years of industrial working experience at international companies. He worked as Visiting Scholar at The Australian National University and The City University of Hongkong in 2005 and 2009, respectively. His research interests include Algorithms, Machine Learning, Data Mining and Information Retrieval.

Feng Xia received the B.E. and Ph.D. degrees from Zhejiang University, China, in 2001 and 2006, respectively. He was a Research Fellow at Queensland University of Technology, Australia, from 2007 to 2008. Before joining Dalian University of Technology (DUT) in Mar 2009, with which he is currently an Associate Professor and PhD Supervisor, Dr. Xia was an Assistant Professor at Zhejiang University. He is on the Editorial Board of several international journals. He serves as General Chair, PC Chair, Workshop Chair, Publicity Chair, or PC Member of a number of conferences. He is also the Guest Editor of several journal special issues. Dr. Xia has (co-)authored one book and over 80 scientific papers. His research interests include cyber-physical systems (CPS), mobile computing, and social computing. He is a member of IEEE and ACM.

A Translation Framework for Executing the Sequential Binary Code on CPU/GPU Based Architectures

Erzhou Zhu, Haibing Guan, Guoxing Dong, Yindong Yang, Hongbo Yang
Department of Computer Science and Engineering & Shanghai Key Laboratory of Scalable Computing and Systems, Shanghai Jiaotong University, Shanghai, China
Email: {ezzhu, hbguan, idalang, yasaka, yanghongbo819}@sjtu.edu.cn

***Abstract*—The method of using DBT (dynamic binary translation) to execute the source ISAs binary code on target platforms has been perplexed by low overhead for many years. GPU as a many-core processor has tremendous computational power. Employing GPU as a coprocessor to parallel execute the hot spot of binary code hold a great promise of substantially reduce the overhead of DBT. This paper presents a novel translation framework for constructing the virtual execution environment aiming at accelerating the process of DBT on CPU/GPU based architectures. With parallelizable parts (hot spots) of binary code and their related information, the framework converts the sequential code into PTX form and executes them on GPUs. Under the framework, we need not to rewrite the source code, and the binary compatibility issues between different GPUs are also resolved properly. Experimental results on several programs from CUDA SDK Code Samples and Parboil Benchmark Suite show that the framework can significantly improve the performance, usually have 10X speedup on average compared to X86 native platforms. Especially, when the scale of input become larger, the performance becomes even better.**

***Index Terms*—GPU, Virtual Execution Environment, Parallelization, CUDA, Dynamic Binary Translation, PTX**

I. INTRODUCTION

Dynamic binary translation (DBT) is a commonly used technology in virtualization area. It converts binary code of one ISA (source binary code) to the binary code of another ISA (target binary code) during the runtime. And the target code will be executed right after the conversion. Performance of DBT system, including the overhead of translating procedure and the execution of target code, has perplexed the researchers for many years. Many optimizations have been used to improve this point, such as translation block chaining, form large translation blocks (superblocks), reordering translated instructions to improve pipeline performance, borrow optimization techniques from conventional compilers. In fact, most present DBT systems can reduce translating costs into an acceptable range. Hence, main attention is laid on bringing down the executive time of target code.

Corresponding author: Haibing Guan (hbguan @sjtu.edu.cn)

Purely software optimization technique is not ideal for improving translation speed, thus software combines hardware optimization technique is a good choice. Although using the accelerating hardware outside of the personal computer could gain good speedup, but modern people emphasizing portability, they couldn't tolerate taking the accelerating hardware all the time. So how to fully use the inside PC component to accelerate the translation process is an inevitable trend.

The programmability and large-scale computing ability of GPU (Graphic Process Unit) has been proved most useful in computation-intensive applications, and there has been considerable interest in general purpose computation on GPUs (GPGPU) [1][2][3]. In many cases, performing general purpose computation in graphic hardware can provide a significant advantage over implementing on traditional CPUs. By this reason, employing GPU as a coprocessor to parallel execute the hot spot of binary code hold a great promise of substantially reduce the overhead of DBT.

There some issues to be resolved on the march of efficiently using of GPU to accelerate the process of DBT. First and the most important issue is to detect the hot spots of the sequential binary code. This issue is a part work of binary analysis [4][5], and has achieved much progress by many frameworks, such as Pin [6], DynamoRIO [7], and Valgrind [8]. One common example of hot spot is the heavily executed loops in the sequential binary code. Through binary analysis, information such as loop body, loop indices, loop bound and the dependence relationships can be obtained. Secondly, with the hot spot and its related information, we must convert the sequential hot spot into parallel fashion. With the recent decades, the technique of auto-parallelization has been significant developed. The strategy of auto-parallelization based on polyhedral model [9][10] has been successfully adapted by many projects for its easily transformable and be able to map the executable codes to multi-core architectures. Thirdly, since CPU/GPU based architectures are heterogeneous platforms, the problem of coordinating the CPU and GPU and enabling them to cooperate in harmony need to be resolved. Fourthly, developers are requested to rewrite the source code. Since traditional compute-intensive

doi:10.4304/jsw.6.12.2331-2340

applications are usually developed by common programming languages, such C and C++, and are compiled into binary codes that use different ISAs from the ones of GPUs. It is obliged to rewrite the hot spot of source binary code, and convert it to the forms of code that are appropriate executed in GPU. The last issue is binary incompatibility [11][12]. It is well known that, GPU hardware are evolving rapidly, this situation poses the problem that the codes that developed and tuned for one generation are not compatible with the next generation. Furthermore, different kinds of GPUs are also incompatible. In contrast with source code incompatible, a much tougher problem is binary incompatibility, which means that the application statically compiled with certain compiler may not work on the platform without the special GPU, let alone another kind of or another generation of GPU.

In order to deal with these issues, we construct and implement a virtual execution environment (GXBit we called) aiming at efficiently executing the sequential binary code in CPU/GPU based architectures. By employing the strategy of dynamic binary translation as well as dynamic-static combined binary analysis, GXBit enables the CPU and GPU to cooperate naturally and therefore applications can be efficiently executed on them. The introduction of GXBit will be described in the follow section.

In this paper, we specifically introduce the core component on constructing the GXBit—the translation framework [29]. The framework first transforms the nested loops within X86 binaries into PTX [13] code, then ports the generated PTX code to CUDA [14]. PTX defines a virtual machine and ISA for general purpose parallel thread execution and could be translated at install time to the target hardware instruction set. Speedup can be achieved via launching GPU to execute PTX code instead of running the corresponding sequential code on CPU.

The contributions of this paper can be categorized as follows:

a) A virtual execution environment (GXBit). GXBit can automatically execute the sequential binary code on CPU/GPU architectures.
b) A translation framework for GXBit. The framework can transform the X86 ISA to PTX ISA, and can also deal with the transfer issue between CPU memory and GPU memory.
c) An intermediate representation (GVInst). GVInst brings the gap between sequential code and parallel code in the process of translation.

II. Overview Of GXBit

Before implementing the translation framework, it is needed to introduce the virtual execution environment (GXBit) that the translation framework works for. Actually, GXBit is a DBT system derives from the multi-sources and multi-targets DBT—Crossbit [28]. However, there are at least three main differences between GXBit and Crossbit. First of all, the execution mode of GXBit is two-phase. The second difference is the parallel parts.

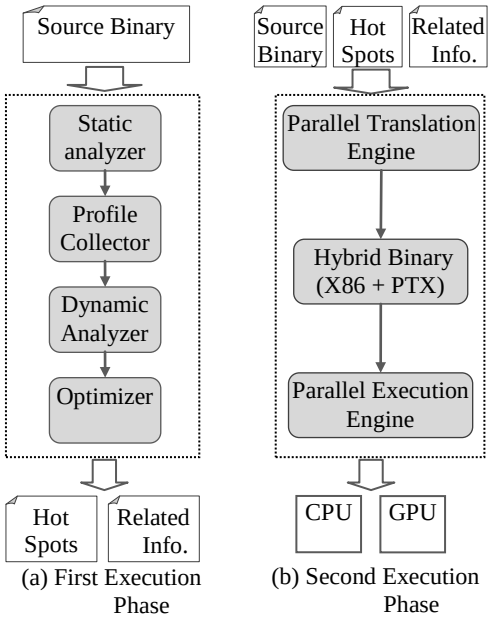


Figure 1. The Workflow of GXBit.

GXBit first extracts the hot spots from source binary code, and then converts these spots to the form that can be recognized and be parallel executed by GPU. Finally, the execution engine of GXBit is also different from Crossbit's, because GXBit needs to execute the paralleled parts on GPU.

Figure 1 is the workflow of GXBit. As the figure describes, the purpose of the first phase is to extract hot spots and their related information from source binary code. Specifically: a) The static analyzer scans the .text section of the input binary to find out all nested loops before the partial execution. These loops are recorded as candidates of hot spots. b) GXBIT starts a partial run for source binary. During the runtime, profile collector inserts additional GVInst after each memory access operation in the VBlocks (the basic translation unit in GXBit) of every candidate. When the translated code is being executed on target, all the accessed memory address can be monitored. c) Once the outmost-level of the candidate nested loop has been executed, the dynamic analyzer uses the monitored information to build a polyhedral model and determines whether the current nested loop has memory dependences between iterations. If there is no dependence, this candidate is regarded as a hot spot and can be parallelized on GPU. d) The optimizer performs certain optimizations to the VBlocks of the hot spots according to specific GPU and dumps all the profiled information and the optimized VBlocks to files.

In the second phase, GXBit utilizes the information collected from the first phase to accelerate the execution of source binary code by porting the hot spots to GPU. This process can be described as: a) GXBIT loads the source binary, profiled information and optimized VBlocks from files, and gets the entry and exit addresses of the hot spots. b) GXBIT starts a whole execution procedure to run the source binary. When the execution flow gets into a hot spot, the parallel translation engine is

triggered to transform the optimized VBlocks to PTX code. Thus forms a hybrid binary, which contains both x86 binary and PTX intermediate code. c) The parallel execution engine will let the hybrid binary run on a CPU/GPU environment, and handle the memory coherence on both architecture.

III. Intermediate Representations

Since raw binary code cannot directly run on GPU, so it is urgently need to design an intermediate representation (IR) that can mask the gap between CPU binary code and GPU's code. It is well known that the code to be executed by GPUs is the kernel function, and the kernel function can be written by CUDA instruction set architecture called PTX. As an assembly form, PTX code can be easily transformed from X86 instructions. On the other hand, the CUDA driver API provides functions to support both loading and executing the PTX code on GPU. So we adopt PTX as the target language of our translation framework. In order to facilitate transforming the hot spots of the binary code to PTX code, we introduce an IR layer (GVInst) to our translation framework.

GVInst is a RISC-like instruction set that provides type safety, flexibility, low-level operations and the capability of representing the critical parts of source binary executable to be translated to the heterogeneous architectures as well as all needed information.

A. Register Architecture

GVInst defines a general-purpose register architecture for virtual machine, it consists of 8 32-bit virtual registers (s0~s7) standing for the 8 general purpose registers in X86 architecture, 8 double precision floating virtual register (f0~f7) corresponding to the floating stack of X86 and infinite 32-bit virtual registers (v0~vn).

B . Addressing Model

GVInst defines RISC style load (LD)/store (ST) instructions to access memory, and the only addressing model it supported is displacement. As we known, an X86 instruction often involves more than one memory access operations. After decoding to GVInst, both explicit and implicit memory operations that represented as LD/ST instructions can be detected.

C. Instruction Format

The basic form of a GVInst instruction is like this:

Optr [.type] Opnd1, Opnd2, [Opnd3], [Opnd4]

Optr tells the function of this instruction. Opnd2, [Opnd3], [Opnd4] are the source operation number, and Opnd1is the destination operation number. In GVInst, most of instructions must have a type field to tell the operand's data type corresponding to type-size specifier on PTX.

As a whole, the instructions in GVInst can be divided into six categories: state mapping, memory accessing, data moving, computing, control transferring and comparing instructions. Table 1 gives the concise description of these instructions.

TABLE I.
THE CATEGORIES OF GVINST

Category	Example and its semantics
State Map (GET, PUT)	GET.type v, s ;Maps a source register (s) to a virtual register (v).
Memory Access (LD, ST)	LD.type v1, (v2.imm) ;Loads the memory data (v2.imm) into virtual register (v1).
Data move (MOV, LI)	LI.type v1, imm ;Stores an immediate data (imm) to a virtual register (v1).
Computation (ADD,SUB,MUL, NOT,AND, …)	ADD.type v1, v2, v3 ;Adds two values in virtual registers(v2,v3).
Control Transfer (JMP,BRANCH, CALL)	JMP V1, imm ;Directly jumps to a memory location (v1, imm).
Comparison (CMP)	CMP.type v1,v2,v3,cc ;Compares the two values(v2, v3) according to the tag (cc), and stores the result (to V1).

The GVInst code is organized in basic block. The term of basic block here means dynamic basic block, which is slightly different from the classic definition in compiler: a basic block is determined by the actual flow of a program as it is executed. It always begins at the instruction executed immediately after a branch of jump, follows the sequential instruction stream, and ends with the next branch of jump. Since loop test will turn to basic blocks containing no statements, extract statements from loop body will be done easily.

IV. The Implementation Of The Translation Framework

As presented in the previous section, in the overall architecture of GXBit, the translation framework standing at the point of receiving the marked hot spots and their related information (written in GVInst), automatically translating these hot spots to the form of code written in PTX and launching the underlying GPU to execute the PTX code. As figure 2 shows, the framework consists of two parts: the translation module, which automatically

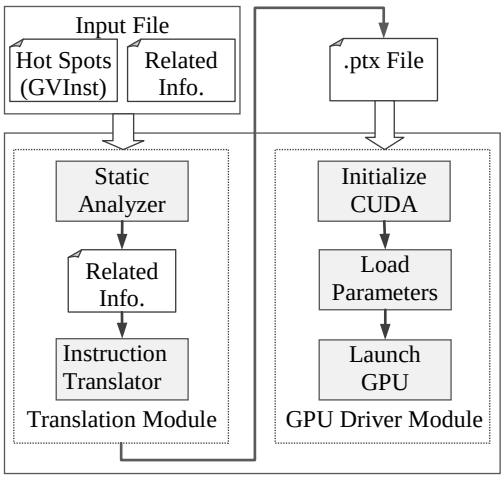


Figure 2. The Translation Framework.

generates PTX file from GVInst; and the GPU driver module, which ports the generated PTX file to GPU and returns the results. The following of this section will describe these components in detail.

A. Translation Module

As a common example of hot spot, in this section, we choose heavily executed loop as the example to demonstrate the procedure of the translation framework. Under this situation, the translation module is responsible for identifying the parts of nested loops, transforming them into PTX code and storing them into a .ptx file.

A) Static Analyzer

The static analyzer first identifies and marks various variables that appear in the loop bodies, then, based on the marked variables, extracts the parallable parts of loop body that really be executed in GPU.

a) Identify and Mark Out the Hot Spots Related Variables

It is has to point out that in GVInst only LD and ST can access memory. We add a filter which is sensitive to LD and ST to monitor the memory access. Once a memory access is detected, a variable checking process of the filter will be triggered. Our framework employs a simple mechanism to name the variables: 1) loop indices are marked as i, j, k...; 2) loop bounds are marked using "bound" prefix, this type of name has an extra field with a numeric character starting with zero. The value is incremented each time when a different loop bound is allocated; 3) other variables are marked using "var" prefix and also have a numeric field as 2).

Figure 3 shows an example for illustration. As soon as the filter detects the LD instruction in line 4 of figure 3(a), the checking process will be trigged. Here s5 stands for EBP in X86 instruction architecture set. So v29(ox0) is a memory address based on EBP with an offset of oxffffffff8. Then we look up the offset value of oxffffffff8 in the variable table, that offered by the input information file shown in the figure 2, to determine whether it is a variable. If it is, the codes will be converted to line 5 in figure 3(b), and the variable is marked as "var1".

```
1.  GET.s32     v6, s5
2.  LI.s32      v29, oxffffffff8
3.  ADD.s32     v29, v6, v29
4.  LD.s32      v1, v29 (ox0)
```

(a) Before marking variables.

```
5.  MOV.s32     var1, v1
```

(b) After marking variables.

Figure 3. Marking variables.

b) Extracting the Statements

The most executed parts of code in loop body should be extracted and be parallel executed on GPU, we call these parts of code as statements. Since for-loop occupy most situations of loops, so in the following we take the for-loop as the major situation. As described in section 3, GVInst employs basic blocks as the basic units to organize its instructions. This feature facilitates the process of extracting the statements from loop body. Under the situation of for-loops, loop tests always appear in the first several basic blocks. This information tells us that these blocks do not contain statements. Therefore, after marking the variables, we regard the basic blocks that only contain loop indices and loop bounds as the loop test part. Following this, we find loop test parts by sequentially examining the basic blocks from the beginning until meet the block with statements. On the other hand, the instructions behind the last ST instruction should be neglected. Considering there is no data dependence in loops, so instructions after the final ST instruction will not influence the results. Take all the situations in to account, the statements can be easily be extracted from the loop body.

B) Translate GVInst to PTX

Prior to introduce the detail of the translation procedure, it is needed to give an overview of the CUDA programming model. It abstracts the thread hierarchy as a grid of cooperative thread arrays (CTAs) which implement CUDA thread blocks. A CTA is an array of threads that execute the same code concurrently in a block. The code to be executed by GPUs is the kernel function. Threads in a block can efficiently communicate with each other through the on-chip shared memory. There is a limitation for the number of threads in a CTA. CTAs that execute the kernel can be batched together into a grid of CTAs. Each thread has a unique thread ID to specify its position is a CTA. In a typical 2D/3D execution domain, the threads in a block have increasing thread IDs along the X direction, and the same thread IDs along the Y and Z directions. Like the thread ID in a CTA, there are CTA IDs in a grid of CTAs and temporal grid IDs in grids as well.

The instruction translation module in figure 2 is responsible for transforming the statements (written in GVInst) into PTX code. Since the source to be translated is binary code, complex optimizations due to their high overhead are unsuitable to our framework. As a compromise, we introduce a simple strategy that translates each GVInst instruction to one or more PTX instructions in the execution sequence.

The following rules are employed to achieve the goal of translation:

a) Mapping Loop Indices to Thread IDs

Taking two nested loops for example, assume that 'i' and 'j' are the loop indices. In the CUDA programming model, 'i' can be viewed as the absolute thread ID along the Y direction, which is equal to (blockIdx.y*blockDim.y + threadIdx.y) in the CUDA. Correspondingly, 'j' is the absolute thread id along the X

direction, which equal to (blockIdx.x * blockDim.x + threadIdx.x). These values could be presented using through predefined, read-only special registers %tid, %ntid, %ctaid, %nctaid, and %gridid [13]. If there is a three nested loops, we can extend this rule into 3D thread hierarchy.

b) Mapping Mechanism of Virtual Registers and Variables

We create a one to one mapping table for recording the register allocation relationship between GVInsts and PTX. According to PTX ISA manual, registers in PTX may be typed (signed integer, unsigned integer, floating point, predicate) or untyped. Register size is also restricted; aside from predicate registers which are 1-bit, scalar register have a width of 8-, 16-, 32- , or 64-bits. Fortunately, our GVInst have defined a field to indicate the operands' type-size specifier. We allocate each variable marked in GVInsts with PTX registers as well. When translating GVInsts to PTX, we will replace the virtual registers and variable with corresponding PTX registers in the mapping table.

c) Transfer Data Values to GPU Memory

Except for loop indices, other variables will transfer their values by PTX kernel function's parameters using CUDA's global memory. We load each parameter value to its corresponding register recorded in the mapping table. In CUDA's memory hierarchy, the global memory is accessible by all threads in a context. It is the mechanism by which different CTAs and different grids can communicate. Comparing with global memory, shared memory is a per-CTA region of memory for threads in a CTA to share data. Although shared memory is much faster, using shared memory should modify the construction of the code which is too complex to handle in binary-level. If all threads could be loaded in one block (less than 512), we will prefer to choose shared memory. To take advantage of shared memory, we should modify the PTX code to ensure that each thread is responsible for load one element.

d) Wrap the Translated Code as a Kernel Function in a .ptx File

The translated code should be completed as a function. In other words, we should complement function heading for the translated code, including function name and parameter list. Furthermore, we need to review the mapping table to statics the registers used in PTX code, and initialize them at the beginning of the function.

B. GPU Driver Module

The workflow of the translation framework turns to the active driver module right after the .ptx file is generated. The GPU driver module is responsible for managing the execution of ported hot spots on GPU. Since the CUDA driver API providers a better interface for handling the assembly-like PTX code than GPU runtime API, our framework adapts the CUDA driver API to implement

```
for (i=0; I < height, i++)
    for (j =0; j < width; j++)
    {
                        …
                        }
```

Figure 4. An example of for-loop.

the GPU driver module. The following steps are used to launch GPU:

A) Initialize CUDA

The purpose of the initialization is to provide an executable environment for running PTX code. For the sake of this purpose, cuInit() and cuCtxCreate() must be called at the first time to initialize the GPU and generate CUDA context respectively. Then cuModuleLoad() and cuModuleGetFunction() are used to load .ptx file and return a function handle. At last, calls cuMenAlloc() to initialize the memory of GPU, and utilize cuMemcpyHtoD() to copy the data from CPU memory to GPU memory. The size of GPU memory to be allocated can be calculated from the input file of loop information. As an example, the memory size to be allocated shown in figure 4 is: width*height*sizeof(float).

B) Load parameters

This step loads the values of formal parameters that appear in PTX functions. The cuParam*() function family is used for loading parameters. In the process of parameter loading, we should consider the following issues: a) the order of loaded values should be the same as the order of the PTX functions' formal parameters. b) The offset of each parameter should be adjusted to meet its alignment. c) Since some variables' value and base address are directly stored in general purpose registers of X86 architecture after optimization, so the 8 general purpose registers should also be loaded as parameters.

C) Launch GPU

At this stage, the first thing is to determine the scale of the computation, such as determining the number of threads per block and the number of blocks per grid. The total number of threads could be calculated by using loop indices and loop bounds. As an example, the total number of threads of the for-loop in figure 4 is width*height. The cuFunSetBlockShape() and cuLaunchGrid() are used to distribute the GPU computing resources and launch GPU

```
cuFuncSetBlockShape ( cuFunction, 16, 16, 1 );
cuLaunchGrid ( cuFunction, ( width +dimBlock.x
– 1 ) / dimBlock.x, \( height + dimBlock.y – 1 ) /
dimBlock.y );
```

Figure 5. Example of setting block and grid with CUDA driver API.

```
/************************************
*Note:
*In this example, we assume that:
* %r2 = blockIdx.y*blockDim.y + threadIdx.y
* %r3 = blockIdx.x*blockDim.x + threadIdx.x
* %r13 = height
* %r14 = width
************************************/
setp.ge.u32 %p0, %r2, %r13;
@%p0 exit;
setp.ge.u32 %p0, %r3, %r14;
@%p0 exit;
```

Figure 6. Instructions Added to avoid redundant threads.

to perform computation respectively. Figure 5 shows the example (appeared in figure 4) of using these API to set block size and grid size.

The total number of blocks should be the integer times of 16, this because the number of 16 can increase the efficiency of GPU memory accessing. Actually, in CUDA programming model, there are 16 threads in a half-wrap. Under this situation, only one data transfer operation is enough to accomplish the task of the 16 threads of a half-wrap to access memory. However, if the number is not the integer time of 16, the threads that have been executed will be exceed the number really needed. To overcome this shortcoming, we insert two exit instructions into the generated .ptx file, as shown in figure 6.

When finished running the kernel function on GPU, we call cuMemcpyDtoH() to return the results back to CPU memory.

V. PERFORMANCE EVALUATION

This section presents the performance evaluation of the translation framework. Table 2 shows the hardware and software configurations of our experimental environment.

We evaluate our translation framework by running 2 applications from CUDA SDK Sample [16] and 2 test case from Parboil Benchmark Suite [17]. As a whole, we examine the effectiveness of the translation framework from two aspects: a) comparing the kernels' performance under different environments; b) deploying the translation framework on a completely virtual execution environment---GXBit.

TABLE II.
HARDWARE AND SOFTWARE CONFIGURATION DETAILS

A) HARDWARE CONFIGURE

CPU	4 * Intel Xeon 5110 clocked at 1.60Ghz (1066Mhz FSB), 4M L2 cache
RAM	8GB, DDR2-667
GPU	NVIDIA GeForce GTX 260, 896MB DRAM, 27 multiprocessors, clocked at 1243MHz

B) SOFTWARE CONFIGURATION

OS	Linux with kernel 2.6.18
Compiler	GCC3.4.3 NVCC2.3
CUDA version	2.3

A. Evaluating the Performance of Kernels

This experiment examines the effectiveness of our translation framework by comparing kernels performance of different programs. We present four experimental results, the two programs that have been chosen from CUDA SDK Sample are Matrix Multiplication and ConvolutionFFT2D, and the other two programs that have been chosen from Parboil Benchmark Suite are MRI-FHD and MRI-Q.

The following tables show the results of running CUDA SDK Sample programs.

Table 3 shows the performance data of running the kernel function in Matrix Multiplication with different input size: 128*128, 512*512, and 1024*1024.

Table 4 shows the results of ConvolutionFFT2D's kernel. This application uses Faster Fourier Transformation (FFT) algorithm to implement a Fourier-based general 2D convolution, which is more efficient than the straightforward. Similar to Matrix Multiplication, we also set three different input data size: 1000*1000, 2000*2000, and 4000*4000.

From the experimental data demonstrated in the above tables, we can conclude that after transforming the hot spots to PTX code by the translation framework: a) The kernel functions achieve consistently much better performance on GPU than the code directly running on CPU (the "native" column in the tables). b) The kernel function that runs on GPU also exhibit better performance than the ones optimized with –O3 flag (the "Native-O3" column in the tables). c) The performance of the experiments exhibits better along with the increasing input scale of data size.

In the experiment, we also compare the performance of running two different versions of the generated PTX code on GPU: the one is generated by our translation framework, and the other one is generated by NVCC (the "NVCC" column in the tables). With a tinge of regret, our translation framework cannot achieve the same performance as NVCC did. The code form of the input

TABLE III.
PERFORMANCE COMPARISON OF MATRIX MULTIPLICATION KERNEL

Matrix Size	Native (ms)	Native –O3 (ms)	Translation Framework (ms)	NVCC (ms)
128*128	31	7	0.44	0.05
512*512	1960	450	21.45	1.61
1024*1024	40620	17400	171.02	12.46

TABLE IV.
PERFORMANCE COMPARISON OF CONVOLUTION FFT2D KERNEL

Image Size	Native (ms)	Native –O3(ms)	Translation Framework (ms)	NVCC (ms)
1000*1000	1570	290	35.43	1.99
2000*2000	6312	1180	141.12	7.95
4000*4000	25240	4690	563.64	33.24

TABLE V.
PERFORMANCE COMPARISON OF CONVOLUTION MRI-FHD KERNEL

Input Size	Native (ms)	Translation Framework (ms)	NVCC (ms)
Small (32_32_32_dataset)	13065	47.97	5.27
Large (64_64_64_dataset)	70340	255.17	23.09

TABLE VI.
PERFORMANCE COMPARISON OF CONVOLUTION MRI-Q KERNEL

Input Size	Native (ms)	Translation Framework (ms)	NVCC (ms)
32	13078	47.86	3.26
64	69912	254.73	11.96

may be the major reason behind this phenomenon. The input of NVCC is the source code, so the PTX code generated by NVCC can efficiently utilize the underlying GPU resources. However, the input of our translation is binary code. Take this reason in to consideration, we still satisfy with the results.

We also employed our framework to generate kernels in Parboil Benchmark Suite: MRI-FHD and MRI-Q. Table 5 and table 6 show the experimental results of running these programs. From these experiments we further prove that the performance our framework works better than that of the native platforms.

Finally, to better understand the achieved performance of our translation framework, we present (figure 7) the comparison of the performance between "native" and "translation framework" that run hot spots (or kernels) respectively. The data shown in the figure is derived from equation (1).

$$Speedup = \frac{Execution\ time\ of\ native\ platform}{Execution\ time\ of\ our\ framework} \quad (1)$$

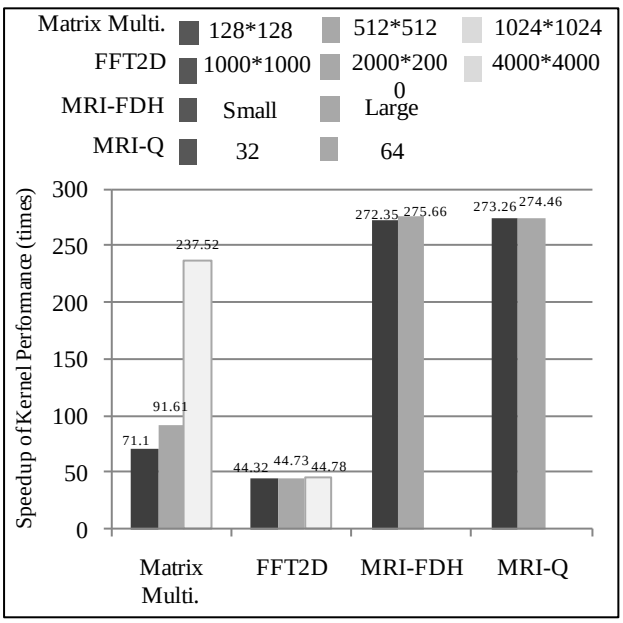


Figure 7. Performance of the kernels gained by the translation framework (Compared to Native Platform).

In equation (1), the execution time of native platform refers to the execution time of hot spots that directly run on CPU platform; the execution time of our framework refers to the execution time of hot spots (or kernel functions written by PTX code) that run on GPU. Since our translation framework can fully exploit the large-scale computing ability of GPU (to parallel execute the hot spots), we gain consistently better performance (up to hundred times of speed up) over the native platform.

B. The Performance of the Framework in Really Environment

In order to demonstrate the feasibility of our translation framework, we evaluate the performance of deploying the framework on a complete virtual execution environment---GXBit. As mention in section 2, GXBit is a virtual execution environment based on CPU/GPU architectures. It is designed for supporting existed binary executable written by sequential language to take advantage of GPU to accelerate the execution of hot spots automatically. Figure 8 shows the performance of the programs running on GXBit over the native ones. The y-axis represents the times of speedups of the programs beyond native. The times of speedups are derived from equation (2):

$$Speedup = \frac{Execution\ time\ of\ programs\ running\ on\ native\ platform}{Execution\ time\ of\ programs\ using\ GXBit} \quad (2)$$

In equation (2), the execution time of native platform refers to the execution time of applications that directly run on CPU platform; the execution time of programs using GXBit refers to the execution time of the applications that run on CPU/GPU based platform. By employing our translation framework, GXBit can fully exploit the large-scale computing ability of GPU. When an application runs on GXBit, the regular process is performed by CPU, and the execution will transfer to

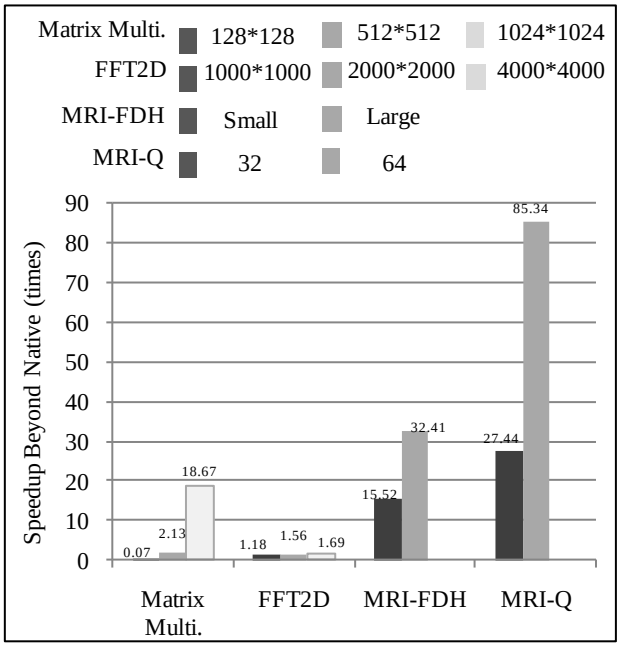


Figure 8. Performance of the programs running on GXBit compared with running on Native.

GPU when a parallel hot spot is detected. The execution will turn back to CPU for the following process right after the GPU have finished execution the hot spot.

The performance would be promoted when the number of the speedup is greater than one. As the figure shows, the performance is improved in most of cases running on GXBit, except for the Matrix Multiplication with 128*128 input data size. In theory, we can gain much performance by employing our framework to port the hot spots to GPU. However, as a virtual execution environment, the binary-level input nature of GXBit is one of reason that slows the process of execution. Additionally, in order to run the applications on CPU/GPU based architectures, GXBit needs extra operations to generate hybrid binary code and transfer data between CPU memory and GPU memory. Therefore, if the performance we gained from GPU cannot compensate for the consumed performance of the above two situations, the overall performance of GXBit would be degraded. This is the direct reason why the speed of running Matrix Multiplication with 128*128 input data size on GXBit is slower than the one running on the native platform.

Figure 8 also shows that the speedups of running ConvolutionFFT2D on GXBit are not ideal even close to the performance of native ones. This because the improvement gained from GPU is eliminated by the overhead of binary translation procedure of GXBit of random initializing the input data array. Actually, GXBit plays a role of binary translator when executing the program except for the parts of kernel functions.

VI. Related Works

The powerful computing ability and the explicit programming environment of GPU have attracted much attention on transforming programs written with other languages to CUDA to obtain more performance improvements. However, the programming environment provided by CUDA is different from the traditional ones. In order to employ the powerful GPU, developers are still needed to rewrite the source code to bring the gaps between the architectures of CPUs and GPUs. For the purpose of avoiding rewrite the source code, many works have emerged to support the CUDA backend. Baskaran [18][25] designed and implemented a transforming framework with an aim to automatically transform affine C programs into CUDA. Lee [19] also developed a compiling framework to complete-automatic transform OpenMP to CUDA. Par4all [20] is a new tool that can translate C and FORTRAN programs to CUDA to accelerate to speed of programs executing. For supporting multi-core architectures, Bondhugula [21] implemented a framework to automatically generate OpenMP parallel code from C programs. However, these works are based on source code and the program analysis techniques based on source code are so mature that they are easier to implement. Considering our translation framework is working on binary-level, so there are many differences from them.

It is also needed to pay much attention on resolving the asymmetric issues produced by the heterogeneous architectures. It is critical to fully utilize the underlying hardware resources on the march of achieving high performance. There are many researches on avoiding the problems caused by asymmetric memory system of the heterogeneous platforms. Gelado [22] gave an asymmetric distributed share memory model for heterogeneous parallel systems. Bratin [23] designed a programming model for heterogeneous X86 platforms. Nathan [24] and Scott [26] did the similar things to bring the gaps of architectures between the accelerators and the host CPUs. However, both of their works are based on their special designed hardware. These special hardware-based interfaces between the bus of host-ends and the accelerators can avoid the problems brought by the heterogeneous memory systems. Yang [27] and Baskaran [15] have derived several methods to resolve the issues of memory optimizing on CUDA. Most of the above works are not limited to CPU/GPU architecture, and the inputs of these platforms are also not limited to binary-level. However, the ideas and the methods behind them have given us much help on designing our translation framework.

VII. Conclusions

In this paper, we presented a novel translation framework for constructing the virtual execution environment with an aim to accelerate the process of DBT on CPU/GPU based architectures. With the input information of binary-level hot spots and their related information, the translation framework can automatically transform the sequential binary code to PTX code, and execute them on GPUs. By introducing the intermediate representation---GVInst, the issues of rewriting source code and the binary compatibility between different GPUs were properly resolved. In the process of translation, by using the mechanisms of identifying and marking variables, our framework efficiently extracted the statements from the loop bodies, then translated these statements into PTX form, and stored them into a .ptx file. In the stage of launching GPU, we employed CUDA driver API other than CUDA runtime API on the reason that the former offers a better level of controlling the assembly-like PTX code. This API also provides us many useful functions for resolving the memory management issue between CPU and GPU. Experiments for benchmark programs have shown that our translation framework achieved better performance than the native ones. Especially, the larger scale of the input data, the higher performance we gained.

In the future, we will consider the following problems: a) the translation mechanisms that affect on the performance should be optimized and be further studied. b) The issue of GPU memory utilization and computation workload distribution should be investigated. c) Extending the framework so as to support the AMD/ATI stream. Finally, we will improve and perfect the framework, and enabling it to support “real-life” situations other than only focusing on benchmarks.

ACKNOWLEDGMENT

This work was supported by The National Natural Science Foundation of China (Grant No. 60970108, 60970107), The Science and Technology Commission of Shanghai Municipality (Grant No. 09510701600, 10ZR1416400, 10DZ1500200, 10511500102), IBM SUR Funding and IBM Research-China JP Funding.

REFERENCES

[1] K.Fatahalian, J.Sugerman, and P.Hanrahan, "Understanding the efficiency of GPU algorithms for matrix-matrix multiplication", Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, 2004, pp.133-137.

[2] N.K.Govindaraju, S.Larsen, J.Gray, and D.Manocha, "A memory model for scientific algorithms on graphics processors", Proceedings of the 2006 ACM/IEEE conference on Supercomputing, 2006.

[3] General-Purpose Computation Using Graphics Hardware. http://www.gpgpu.org/.

[4] Cristina Cifuentes and K. John Gough, "Decompilation of Binary Programs", SOFTWARE: PRACTICE AND EXPERIENCE, VOL. 25(7), 1995, pp.811-829.

[5] Tipp Moseley, Daniel A. Connors, Dirk Grunwald, Ramesh Peri, "Identifying potential parallelism via loop-centric profiling", Proceedings of the 2007 International Conference on Computing Frontiers, 2007, pp.143-152.

[6] C. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, Vijay Janapa Reddi and K. Hazelwood, "Pin: building customized program analysis tools with dynamic instrumentation", Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation, 2005, pp.190-200.

[7] Nicholas Nethercote and Julian Seward, "Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation", Proceedings of ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation, 2007, pp.89-100.

[8] Derek Bruening, Timothy Garnett, Saman Amarasinghe, "An Infrastructure for Adaptive Dynamic Optimization", International Symposium on Code Generation and Optimization, 2003, pp. 265-275.

[9] C. Ancourt and F.Irigoin, "Scanning polyhedral with do loops", Symposium on Principles and Practice of Parallel Programming, 1991, pp.39-50.

[10] U. Bondhugula, A. Hartono, J. Ramanujan, and P. Sadayappan, "Apractical automatic polyhedral parallelizer and locality optimizer", Proceedings of the 2008 ACM SIGPLAN conference on Programming language design and implementation, 2008, pp.101-113.

[11] Nathan Clark, "Why Should I Rewrite My Software When Dynamic Compilation Can Be Good Enough", Workshop on Software Tools for Multi-Core Systems, 2008.

[12] Nathan Clark, Jason Bolme, Micheal Chu, Scott Mahlke, Stuart Biles, and Krisztian Flautner, "An Architecture Framework for Transparent Instruction Set Customization in Embedded Processors", International Symposium on Computer Architecture, 2005, pp. 272-283.

[13] "PTX: Parallel Thread Execution ISA", Version 2.1, 2010.

[14] "NVIDIA CUDA Programming Guide", Version 3.1, 2010.

[15] Muthu Manikandan Baskaran, J. Ramanujan, Sriram Krishnamoorthy, "Automatic Data Movement and Computation Mapping for Multi-level Parallel Architectures with Explicitly Managed Memories", Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming, 2008, pp.1-10.

[16] Victor Podlozhnyuk, "FFT-based 2D convolution", NVIDIA CUDA Sample Documentation, 2007.

[17] "Parboil benchmarkSuite", http://impact.crhc.illinois.edu/parboil.php.

[18] Muthu Manikandan Baskaran, J. Ramanujan, "A Compiler Framework for Optimization of Affine Loop Nests for GPGPUs", Proceedings of the 22nd annual international conference on Supercomputing, 2008, pp.225-234.

[19] Seyong Lee, Seung-Jai Min, Rudolf Eigenmann, "OpenMP to GPGPU: a compiler framework for automatic translation and optimization", Proceedings of the 14th ACM SIGPLAN symposium on Principles and practice of parallel programming, 2009, pp.101-110.

[20] Par4All, http://www.par4all.org/

[21] Uday Bondhugula, J.Ramanujam, P.Sadayappan, "PLuTo: A polyhedral automatic parallelizer and locality optimizer for multicores", http://pluto-compiler.sourceforge.net.

[22] Isaac Gelado, Javier Cabezas, John Stone, Sanjay Patel, Nacho Navarro and Wen-mei Hwu, "An Asymmetric Distributed Shared Memory Model for Heterogeneous Parallel Systems", Proceedings of the fifteenth edition of ASPLOS on Architectural support for programming languages and operating systems, 2010, pp.347-358.

[23] Bratin Saha, Xiaocheng Zhou, Hu Chen, Ying Gao, Shoumeng Yan, Mohan Rajagopalan, Jesse Fang, Peinan Zhang, Ronny Ronen, Avi Mendelson, "Programming Model for a Heterogeneoous x86 Platform", Proceedings of the 2009 ACM SIGPLAN conference on Programming language design and implementation, 2009, pp.431-440.

[24] Nathan Clark, Amir Hormati, and Scott Mahlke, "VEAL: Virtualized Execution Accelerator for Loops", 35th International Symposium on Computer Architecture (ISCA), 2008, pp.389-400.

[25] Muthu Manikandan Baskaran, J. Ramanujan and P. Sadayappan, "Automatic C-to-CUDA Code Generation for Affine Programs", 19th International Conference on Compiler Construction, 2010, pp.244-263.

[26] Hyunchul Park, Yongjun Park, and Scott Mahlke, "Polymorphic Pipeline Array: A Flexible Multicore Accelerator with Virtuzlied Execution for Mobile Multimedia Applications", Procedings 42nd International Symposium on Micro architecture, 2009, pp.370-380.

[27] Yi Yang, Ping Xiang, "A GPGPU Compiler for Memory Optimization an Parallelism Management", Proceedings of the 2010 ACM SIGPLAN conference on Programming language design and implementation, 2010, pp.86-97.

[28] Yindong Yang, Haibing Guan, Erzhou Zhu, Hongbo Yang, Bo Liu, "CrossBit: A Multi-Sources and Multi-Targets DBT", The First International Conference on Cloud Computing, GRIDs, and Virtualization, 2010, pp.41-47.

[29] Guoxing Dong, Kai Chen, Erzhou Zhu, Yichao Zhang, Zhengwei Qi and Haibing Guan, "A Translation Framework for Virtual Execution Environment on CPU/GPU Architecture", the Third International Symposium on Parallel Architectures, Algorithms and Programming, 2010, pp.130-137.

Erzhou Zhu is currently a Ph.D student at Shanghai Jiao Tong University, China. He received the M.S. degree and B.S. degree in Computer Science and Technology in Anhui University, Anhui, China, in 2004 and 2008 respectively. His research interests include virtual machine, binary translation and computer architecture.

Haibing Guan received his Ph.D. degree in computer science from the TongJi University (China), in 1999. He is currently a professor with the Faculty of Computer Science, Shanghai Jiao Tong University (Shanghai, China). His current research interests include, but are not limited to, computer architecture, compiling, virtualization and hardware/software co-design.

Guoxing Dong is currently a Master Degree Candidate student at Shanghai Jiao Tong University, China. He received the B.S. degree at Shanghai University in 2008, China. His main research interests are in binary translation and CPU-GPU Co-Processing.

Yindong Yang is currently a Ph.D student at Shanghai Jiao Tong University, China. He received the M.S. degree at School of Computer, Electronics and Information from Guangxi University in 2007, China. In 2004 he received his BS degree at School of Information and Technology from Jiangnan University, China. His main research interests are in virtual machines, computer architecture and compiling.

Hongbo Yang is currently a Ph.D. student at Shanghai Jiao Tong University, China. He received the M.S. degree in 1995 and received his B.S. degree in 1998 at Institute of Airforce Meteorologyity, China. His main research interests are in virtual machines, computer architecture and compiling.

A Dynamic-Static Combined Code Layout Reorganization Approach for Dynamic Binary Translation

Haibing Guan, Erzhou Zhu, Kai Chen, Ruhui Ma, Yunchao He, Haipeng Deng and Hongbo Yang
Department of Computer Science and Engineering & Shanghai Key Laboratory of Scalable Computing and Systems
Shanghai Jiaotong University, Shanghai, China
Email: {hbguan, ezzhu, kchen, ruhuima, ventureheart, denghipson, yanghongbo819}@sjtu.edu.cn

***Abstract*—Dynamic binary translation (DBT) has attracted much attention as a powerful technique for the runtime adaptation of software among different ISAs. It offers unprecedented flexibility in the control and modification of a program during the runtime. However, its inherent high overhead has perplexed researchers for many years. In order to reduce the overhead of DBT, this paper presents a dynamic-static combined approach to reorganize the layout of software cache. Under this approach, we first employ an emulating execution to collect the profile information and the translated target code. Especially, the path of execution flow will be tracked. In the static phase, based on the profile information collected in the previous stage, we first use the method of code replicating to build the traces, and then reorganize the layout of the target code by putting the hottest traces at the top of the software cache. Because of exact prediction and improved locality, the execution stream will concentrate on a small area with less control transfer. This approach can greatly reduce the overhead of DBT on the condition that the program runs repeatedly. Experimental results on executing the SPEC 2000 benchmarks show that our approach can reduce more than 30% run time on average.**

***Index Terms*—Dynamic binary translation, profile information, static optimization, code replication, trace building**

I. INTORDUCTION

Dynamic binary translation (DBT) has attracted much attention as a powerful technique for dynamically adapting software among different ISAs [1]. It offers unprecedented flexibility in the control and modification of a program during the runtime. The technique of DBT can be used in emulating an ISA to a new ISA, monitoring and optimizing performance at runtime, providing resource protection and management, virtualizing resources, and detecting security attacks and so on. In the past decades, a lot of dynamic binary translators have been designed for different goals: IA-32 EL [2] and Aries [3] can migrate applications cross different ISAs; Pin [4], Dynamo [5], Valgrind [6] and HDTrans [7] can detect the behaviors of the programs as well as optimize the programs at runtime; Shade [8] and QEMU [9] use DBT to speed up architecture simulations; DAISY [10] is designed for creating a virtual execution environment from a totally different architecture at ISA level; ADORE [11] is designed as a dynamic binary optimization system.

However, the inherently high overhead feature of DBT has perplexed many researchers for many years. Generally speaking, the overhead incurred in DBT can be divided into two parts: the overhead of translating procedure and the overhead of executing the target code. As a matter of fact, most of the present DBT systems can reduce the translating cost to an acceptable range. Hence, the main attention is laid down on the reducing execution time of target code. In order to reduce the time consuming of executing the target code, many optimizations has been proposed, such as translation block chaining, forming larger translation blocks (superblocks), reordering translated instructions to improve pipeline performance, and borrowing optimization techniques from traditional compilers. In fact many binary-code-specific optimizations are seriously depend on the profile information gathered during the runtime. The richness and correctness of profile information can directly determine which kind of optimization can be implemented and the extent of its efficiency. Profiling is a process for dynamically collecting program information (instructions and data statics) that is used to guide the optimization during the translation process. However, this inevitably leads to performance losing, especially the complex one.

By introducing the static analysis stage into dynamic binary translation, the overhead of profile and dynamic optimizing are reduced remarkably. Under this scenario, the complex profile is collected by the first emulating phase. Then efficient optimization algorithms based on profile information collected by the previous stage are available. The overhead of the subsequent executions will be small, since they can directly execute the optimized target code generated by the static optimization stage. Generally speaking, the method of combines the dynamic translation with the static analysis at least has the following merits: firstly, it can spare the translation time.

Corresponding author: Erzhou Zhu (ezzhu@sjtu.edu.cn)

doi:10.4304/jsw.6.12.2341-2349

The process of translating source code to target consumes a substantial part of time. Once the source code is translated at first run, then the target code will be saved in special files which can be directly loaded at the next runs. Secondly, profiling overhead will be eliminated except for the first run. If all the optimizations are performed statically, the profiling process is not needed anymore. Lastly, complex optimizing algorithms are available now. Since many optimizations are carried out statically instead of at runtime, complex algorithms which are not appropriate dynamically are available now without warring about overhead.

Actually, the approach of employing static phase to aid dynamic binary translation assumes that the execution profile of each block in the profile phase (initial emulating execution phase) is representative of the block throughout its lifetime. In particular, a region is selected for optimization with the assumption that it infrequently takes its side exits and is thus candidate for advanced optimizing. If these assumptions are not hold, however, the performance of the program will be suffered [12].

In DBT systems, the native code that CPU needs to execute is stored in software maintained cache. By reorganizing the layout of software cache, the overhead of dynamic binary translators are reduced remarkably. Once the target code in software cache is reframed properly, hot code will be gathered together and organized properly. Therefore, the execution stream will concentrate on a small but hot area with less happening of control transfer. As a whole, the method of reorganizing the layout of software cache can: (1) decreases the execution of jump instructions, (2) makes the pipeline going on with fewer interruptions, (3) reduces physical cache miss rate, (4) cuts down TLB access miss rate, and (5) decreases the page faults.

This paper presents a new dynamic-static combined approach for reorganizing the layout of software cache in DBT system. This approach is based on the static-integrated optimization framework appeared in [30], and the DBT system that our approach is going to optimize is Crossbit [29]. As a whole, the approach contains three stages to fulfill its optimization target. In the initial phase, we employ the emulating execution to collect profile information and the translated target code. Especially, the path of each execution flow will be tracked, which is used to build the trace in the static analysis phase. In the static analyzing phase, based on the profile information and target code that are collected from previous stage, we first use the method of code replicating to build traces, and then reorganize the layout of the target code by putting the hottest trace at the top of the software cache. At the last stage, i.e. the subsequent executions, due to the profile information and the optimized target code are all available the overhead will be remarkably reduced. Different from our original approach [27], this approach introduces a replicate-based trace building algorithm in the static analysis phase. By code replication, more hot traces will be detected and generated in the memory space, through which programs could execute more continuously and therefore performance promotion will be achieved.

The remainder of this paper is organized as follow. Section II introduces a brief overview of the DBT system (Crossbit) that out approach is going to optimize. Section III presents the detail implementation of the approach, which includes the overall workflow of our approach, traces building, code replicating and code layout reorganizing. Section IV gives the performance evaluations. Some related works are presented in section V. At last, we conclude our work.

II. BACKGROUNDS

Crossbit is the target DBT system that our approach is going to optimize. It is designed and implemented as a multi-source architectures and multi-target architectures dynamic binary translator, which aims at fast migrating existing executable source code from one platform to another alien target platform with lower cost. Until recently, it has fully or partially supported source platforms including SimpleScalar, IA32, MIPS, SPARC, and supported target platforms such as IA-32, Power PC and SPARC. The operating system that Crossbit support is Linux. In order to support code translation among multi-sources and multi-targets better, a new Intermediate Instruction set—VInst [31], which is independent of any specific machine instructions, has been introduced. Unlike many other existing DBTs which directly translate the binary code of one instruction architecture (ISA) to another ISA, Crossbit first converts source binary code to VInst specifications and then transforms them into target platform code, using a granularity of a basic block (BB) as the basic unit of translation. The detail introduction of Corssbit is described in [29].

Software-managed code cache, also called software cache or code cache for short in Crossbit, is very critical in Dynamic Binary Translator. It usually occupies an area of main memory and stores translated native code, making the translator reuse the native code to avoid the overhead of retranslation during the whole execution. Therefore, it can remarkably improve the performance of DBT system. In order to remove noise in performance comparison and analysis, we suppose the size of Software Cache is unbounded. It can store all of the translated code without any replacement policy. Then in Crossbit, the running time of execution over translated code takes more than 97% on the whole according to the benchmark of SPEC2000, while the cost of initialization, translation and optimization is trivial.

In Crossbit, many techniques are employed to improve the quality of translated code in an effort to develop the performance:

(1) Linking between blocks to reduce the incidence of returns to Crossbit
(2) Changing indirect jump into several compare and direct jump in translated code to reduce the context switch to Crossbit when executing the translated code.

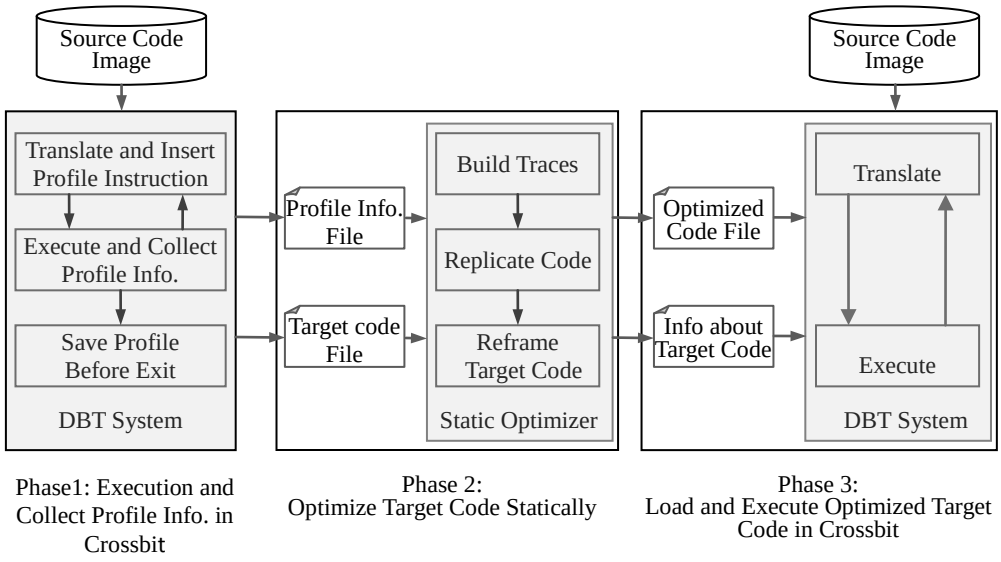


Figure 1. The Workflow of Our Approach

(3) Building superblock according to profile.
(4) Redundant code elimination.

These conventional techniques can greatly develop the performance of Crossbit, making it comparable to QEMU. To do further optimization, we present the method of dynamic-static combined code layout reorganization in software cache.

III. IMPLEMENTATIONS

A. Overview

As Fig.1 demonstrates, the overall workflow of our approach can be divided into three phases: dynamically collects profile information at the first emulating execution phase, performs profile-directed optimizations (trace building and code layout reorganizing) in static analysis phase, and loads the target code that has been optimized offline for the subsequent executions.

A) Emulating Execution

In this phase, the DBT executes the source program dynamically. Instrumentations are deployed to collect the profile information. Profile information contains the execution times of each block, the incoming blocks and outcoming blocks of each block, the time of each jump direction, and so on. Especially, the path of each execution flow will be tracked, which is used to build traces in the static optimization stage. After the execution of the source image, the profile information will be saved in profile file. Specially, in order to spare the execution time of the subsequent execution, the translated target code will also be saved (in target code file) for the purpose of optimizing in the static analysis phase.

B) Static Analysis

Since it is needn't taking much consideration of analyzing overhead in the static analysis phase, sophisticated optimizations can be deployed. We can perform the optimizations than can take full advantage of rich profile information, such as conditional branch directions prediction, inline and build superblocks according to the execution times of current blocks and its hottest outgoing edges. In this paper, we focus on trace building based on the method of code replication and translated code layout reorganizing. In this approach, hot traces are identified by their head blocks, and the granularity of the code to be reorganized is basic block. At the end of the static analysis stage, the hottest traces are placed at the top of the software cache, while the coldest ones are placed at the bottom.

C) Subsequent Execution

In this stage i.e. the subsequent executions, the operations of loading (profile file and target code file) and initializing are carried simultaneously. The main work flow of DBT is not changed, which means that the binary source image should be loaded as normal. The difference is that the target code is directly loaded from target file instead of been translated from source image. When execution starts, DBT tries to find the optimized code other than source code. If the target code that should be executed now has existed, DBT just executes it directly. Otherwise, DBT will translate the source code as normal. This approach can greatly reduce the overhead of DBT on the condition that the program runs repeatedly.

In the overall workflow of our approach, stage 2 (i.e. the static optimization stage) is the most important one. It first takes the profile information and the translated code as the input, then builds traces by employing the method of code replicating, at last, reorganizes the layout of code cache by putting the hottest traces at the top area.

B. Trace Building and Code Replicating

Hot trace building plays an important role in enhancing the performance of dynamic binary translation. As a matter of fact, in most cases only 10% of code takes 90% of execution time of the whole program. Hot traces can promote the code position to make better the locality of the code, and therefore programs can achieve a better performance. In conventional, there are many traces in a

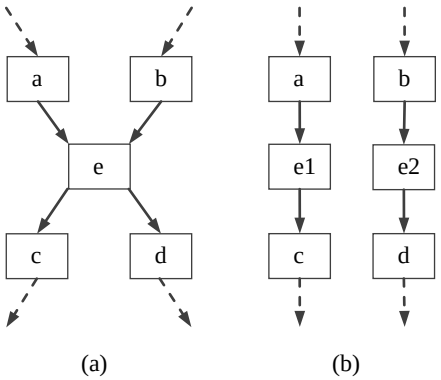


Figure 2. An Example of Code Replication.

program, control transfers are frequently occurred from one to another. In DBT or other dynamic systems, trace building is a different issue, which must take a balance between the runtime overhead and efficiency. So how to enhance the quality of the trace fragment, which is a sequence of frequently-executed basic blocks in the memory, is quite important. At present, many trace building algorithms have been developed which can detect hot traces accurately. However, the high overhead make them less useful when applied in dynamic systems. In additional, there is a common situation in which a block is shared by many traces, only one of them is maintained intact while the others will be truncated into many short trace fragments. These short trace fragments will be produced in memory space and the instruction locality will get worse.

To overcome the problems mentioned above, our approach employs a different strategy. In this strategy, we statically build traces based on the profile information that has been collected from the dynamic emulating execution stage, and employ the method of code replication to resolve the problem that one block is shared by many traces. By employing the dynamic-static combined method of trace building, the unbearable high overhead of DBT is turned away. It is need to be mentioned that, in this situation, the process of trace building and code replicating are on the level of logical, the works that really happen are in reorganization stage.

A) Trace Building

Since the translated code (organized as basic blocks) and its corresponding profile information are dynamically collected from the emulation stage, the process of trace building can be easily performed during the static phase. We start our approach by traverse the files that contain profile information and target code, and then employ the following two steps to build traces: the first step is to identify the trace head. The head block of a trace is determined by the Begin-Threshold, a threshold value that we predefined. When the execution time of a basic block exceeds the Begin-Threshold, it will be set as a trace head. Secondly, when the trace head is identified, the trace building procedure will be called to build trace. As we know, when one block becomes hot, the other blocks surrounding this hot block also seems to be hot. Consequently, the exit block of the current block that has the maximum execution time will be selected as a part of the current trace and also be labeled as “trace-men”. When the maximum execution time of the exit block is blow the Finish-Threshold (usually a multiple of the Begin-Threshold) or the block to be appended is already labeled as “trace-men” the trace building procedure will be closed, and thus the trace fragments emerged.

B) Code Replicating

The method of trace building in section A) will produce many trace fragments. In that situation, when a block is shared by different traces, only one of these traces is maintained intact, while the others will be truncated into many short trace fragments. These short fragments will be placed in the memory and the instruction locality will get worse. To handle this problem, we present a method of replicating of the very block that been shared by many different traces, and thus improve the instruction locality as well as memory continuity.

Fig.2 gives a simple example of code replication. Fig.2 (a) supposes the situation that both block a and b are jump to e, and then e passes the execution to c and d respectively. In this case, four traces might be constructed: ⋯a→e→c⋯,⋯a→e→d⋯,⋯b→e→c⋯,⋯b→e→d⋯. This way, the execution stream will be interrupted more frequently, and exhibits bad locality characteristics. According to our approach, we make a copy of block e when the second trace is built, like ⋯b→e2→d⋯. By doing this, the number of trace fragments is reduced, and the length of trace fragments is increased. Then two longer traces will be constructed (as Fig.2 (a)) which display a good instruction locality and allow the execution stream to run more continuously. However, this method also introduces a side effect of memory expansion, this side effect will be detail discussed in section IV.

C) Trace-Table

Since the works performed in this section are all on the level of logical. We just record the trace information and the replicated blocks. The works that really happen are in reorganization stage. In order to record this information, we maintain a Trace-Table that records the trace information, such as the SPC of the head block (identifies the trace) and its corresponding execution time (identifies the frequency of the trace), the SPC of the remainder blocks of the trace and so on.

C. Code Layout Reorganization

As mentioned previously, the scale of the software cache in Crossbit (the very DBT system our approach is applied to) is been set unbounded. By doing this, it is only needs a little time to initialize and translate. Additionally, reorganizing code layout of software cache can improve the performance on the ground that the execution stream will be more approximate to its control flow graph after reframing. On the target code in software cache is reframed properly, hot code will be gathered together and be well organized. Because of exact prediction and improved locality, the execution stream

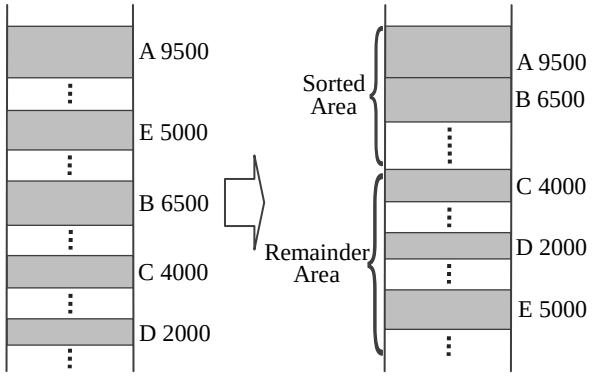


Figure 3. Sorting the head blocks according to their execution frequency.

will concentrate on a small area with less control transfer. The work of code layout reorganization is followed by the work of trace building and code replicating.

A) Sort Target Traces and Blocks

Prior to perform the work of code reorganizing, is has to sort the target traces and the remainder target blocks (scattered block we called) that have been generated previously. Since traces are identified by their head blocks, we just need to sort the head blocks. The right figure in Fig.3 displays the code that has been sorted from the left one.

B) Reorganize Code Layout

Since a trace is identified by its head block. If the head block of a trace is placed on the top of code cache, then the remainder blocks of the trace will be placed adjacent to their head blocks. Fig.4 demonstrates the change of the code layout. Supposes A, E, C are built into a trace, A is the head block of this trace, and A is the hottest block, then all of them will be placed on the top in spite of block B is more hotter than most of blocks in this trace.

IV. Performance Evaluations

In this section, we first carry out some experiments on comparing the overall performance of our approach with the original version of Crossbit (the DBT system that our framework based from) and QEMU (a fast and portable open source DBT system). Then the experiments on verifying the priority (compared with original Crossbit)

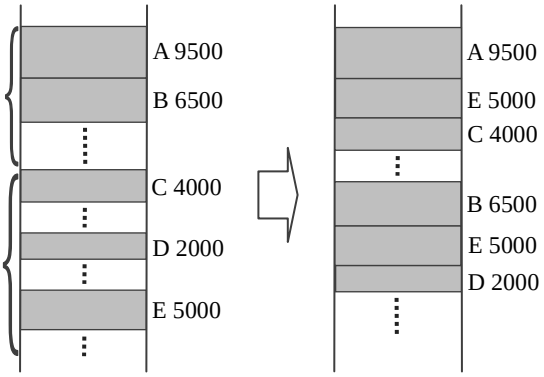


Figure 4. Reorganizing the Layout of Software Cache

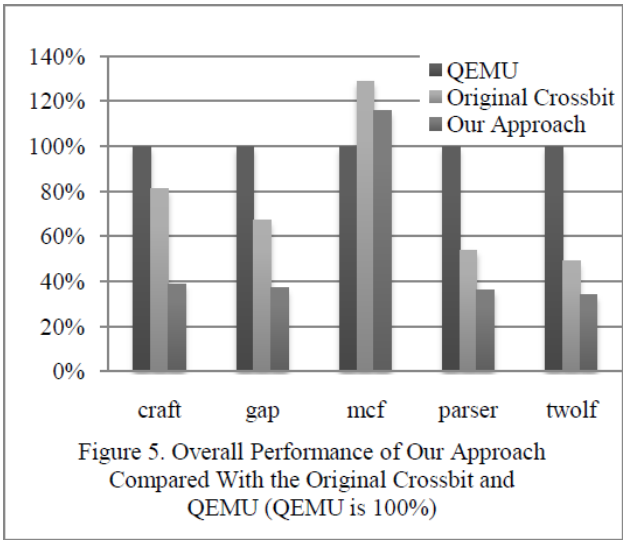


Figure 5. Overall Performance of Our Approach Compared With the Original Crossbit and QEMU (QEMU is 100%)

of our approach are performed. At last, we emphasize the side effect of our approach that incurred by code replication. We carry out these experiments on an Intel Pentium 4 dual-core machine, where each core has a 2.8GHz Pentium with an 800 MHz processor bus, 32KB of L1 cache and 1.5MB L2 cache. The machine's memory system uses a 533MHz bus with 1.5GB of dual interleaved DDR SDRAM memory. We use the SPEC CPU2000 benchmarks as the test suite. Some of the benchmarks cannot run successfully on our approach, which might due to the base Crossbit lacking of complete support for all Linux System calls, or there are some errors in address relocation and linking of exits stub after code layout reframing, our team still deals with these issues now. So the comparisons and analysis are based on these programs that can be right executed on our approach.

A. Overall Performance Comparison

Fig.5 shows the overall comparison of performance among our approach, original version of Crossbit and QEMU [9] (a fast and portable open source DBT system). When compared with original version of Crossbit, our approach displays consistently better performance than that of Crossbit for all of the tested benchmarks. And also outperforms of QEMU for nearly all the tested benchmarks. The overall executing time of our approach is reduced by 34% on average relative to the original version of Crossbit, and more than 50% to QEMU. The better performance we gained largely due to static phase optimization and partly because of directly loading optimized target code to the following execution.

Under our approach, fewer blocks are needed to be translated than the original Crossbit. Since the size of software cache of our approach is set unbounded, there would be no retranslation. Experimental statistic also shows that the overall consumption of initialization, optimization and translation are no more than 3%. So the reduction of the execution time, compared with original Crossbit, is about 31% in SPEC 2000. When compared with QEMU, our approach is much faster except for the mcf benchmark, the reason of this defect is described in [29].

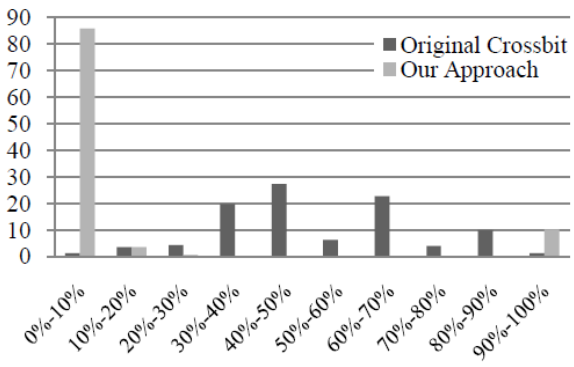


Figure 6. Code Execution Frequency Distribution.

The following two experiments are performed to verify the reasons of high performance we gained from static optimizations.

A) Code Execution Mainly Concentrates on A Few Hot Code

It is well known that when a program is executed, the code execution frequency is often not evenly distributed. A small portion of the hot code usually occupies most of the execution time, while a large portion of other code is to cover all eventualities but executed little. So when the execution focuses on a small portion of the hot instructions, the overhead will be decreased. Fig.6 gives the experiment's results to show the code execution frequency distribution our approach and original version of Crossbit. From the results, we can conclude that the code execution frequency is evenly distributed under original Crossbit. However, under the schema of our approach, more than 85% execution time mainly concentrates on 10% hot code.

B) Less Control Transfer Occurrence

Longer traces can greatly reduce the occurrence of control transfer. Less transfer control occurrence in the same program can enhance the accuracy of instruction pre-fetch, reduce interruption of pipeline, and finally introduce better performance. Fig.7 and Fig.8 show the direct jump occurrence and indirect jump occurrence respectively. In the figures the jump occurrence in original version of Crossbit is supposed as 100%. From the figures we can conclude that our approach can greatly reduce the occurrence of control transfer no matter it is a

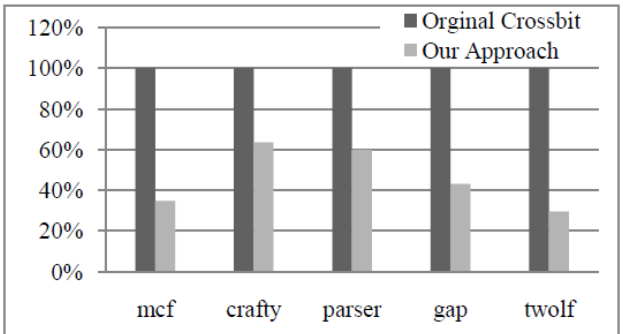


Figure7. The occurance of direct jump (Original Crossbit is 100%)

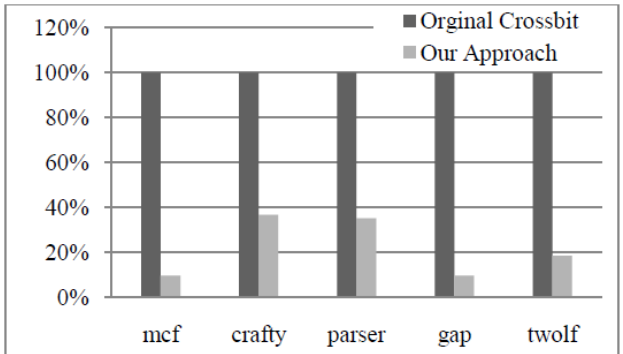


Figure 8. The occurance of indirect jump (Orginal Crosssbit is 100%)

direct jump or an indirect jump. This because our approach can find hot spots and hot traces of the program, and then build longer traces to fulfill loop without transfer control change.

B Side Effect of Our Approach

Lastly we talk about the side effect of code replication: memory expansion. Fig.9 plots the memory expansion ratio (MER) of code replication. It shows the maximum MER of nearly 6% when running the crafty test case, and an average expansion ratio of 2.25%. Particularly, in the case of mcf, we get a performance promotion of 34%, with only a memory expansion of less than 2%. We can find that this code replication is quite suitable for mcf to enhance performance. On the situation of crafty, we can't ignore the memory expansion sacrificed to performance promotion.

V. Related Works

For the purpose of gaining better performance, the static process has been adapted in many binary translation systems. FX!32 [13] is a profile-directed binary translator, which combines emulation (first run) and binary translation (subsequent runs) techniques. It possesses a database to accommodate the target code, which is generated by the background emulator according to the online profile information. Then the target code is available at the subsequent runs of the source image.

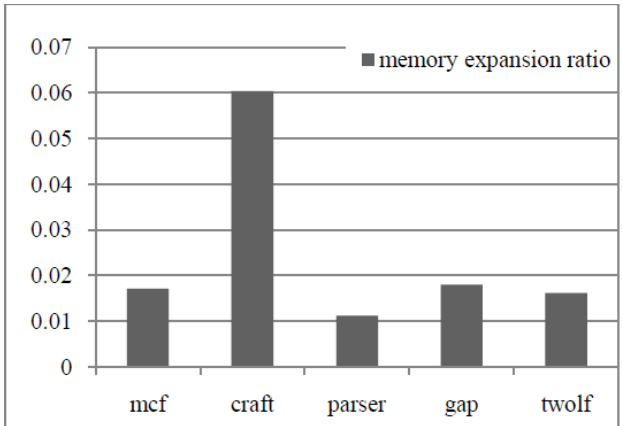


Figure 9. Memory Size Expansion Ratio (compare with original Crossbit)

IA-32 Execution Layer (IA-32 EL) [2] is a two-phase dynamic binary translating software designed for supporting IA-32 applications on Intel Itanium family systems. In IA-32 EL, the first phase (also called cold code translation) is designed to be fast, with minimal optimizations and overhead and uses instrumentation to identify hot spots. The second phase (also called hot code translation), retranslates and further optimizes the hotspots identified by the cold translation. Metadata driven DBT [14] proposes a novel method of passing performance critical information to dynamic binary translator through metadata. By metadata, the dynamic binary translator is able to perform aggressive optimizations to generate high quality code. This approach behaves well if the source programs (written in high level language) are available.

Static process has also been used in conventional compilation systems. GCC [15] requires the programs to be built and executed twice. In the first time, the applications are compiled to generate profile information. And the second time, the applications are compiled and use the profile information that was generated by the first execution.

Efficient software cache Management is an important mean to improve the performance of DBT. In [16] and [17], the authors discussed the replacement policy of code cache. They compared several strategies of code cache replacement policy and presented the appropriate granularity of code cache eviction size, trying to protect more hot code from swapping out when the software cache is limited. Moreover, in an effort to cut down the memory consumption of code cache, they elaborated several methods [18] to reduce exit stub that is used to handle the egress of control transfer. [19], [20] and [21] presented the schemes to keep the software cache consistent between the code cache and the original code, to keep it transparent to user and operating system, to dynamically bound code cache size to match the current working set of the application when it is running, and to make software cache becoming process-shared when this merit is necessary.

For most DBTs, the trace is the unit of choice of function and method. Traces are found to perform well and be easy to translation [22]. Many trace detecting methods have been employed to improve the performance. Next-Executing-Tail (NET) is a popular trace selection algorithm [23]. And it attempts to select traces that begin at loop headers by considering only instructions that are the targets of backward branches or exits from existing traces [24]. But it cannot span inter-procedural cycles and the nested loops. Mueller and Whalley [25] have applied code replication to avoid jumps in the program and improve instruction locality. Measurements taken from a variety of programs showed that not only the number of executed instructions decreased, but also that the total cache work was reduced (except for small caches) despite increases in code size. Adreas Krall [26] presents a kind of code replication techniques to improve the accuracy of semi-static branch prediction scheme. It uses profiling to collect information about the correlation between different branches and about the correlation between the subsequent outcomes of a single branch.

VI. Conclusions And Future Work

As a power technique for the runtime adaption of software among different ISAs, dynamic binary translation has gained much attention. It offers unprecedented flexibility in the control and modification of a program during the runtime. But these attractive features are confined by substantial overhead, especially in the situations that translate among totally different architectures. In this paper, we tried to offset this drawback by introducing a dynamic-static combined approach to optimize the dynamic binary translators. Under this approach, we first employ the emulating execution stage to dynamically collect the profile information and the translated target code. Especially, the path of each execution flow will also be tracked. In the static analysis phase, due to the platitudinous profile information and target code, we employed the method of code replication to build traces, and then reorganized the layout of the target code by putting the hottest trace at the top of software cache. By implementing our approach, the execution flow was concentrated on a small area. Experimental results had shown that our approach reduced more than 34% runtime on average at the expense of 2% memory expansion ratio on average.

In the future, we will employ more algorithms to improve the runtime overhead as well as reduce the memory size expansion ratio.

Acknowledgment

This work was supported by The National Natural Science Foundation of China (Grant No. 60970108, 60970107), The Science and Technology Commission of Shanghai Municipality (Grant No. 09510701600, 10ZR1416400, 10DZ1500200, 10511500102), IBM SUR Funding and IBM Research-China JP Funding.

References

[1] Jinpyo Kim, Wei-Chung Hsu, Pen-Chung Yew, “COBRA: An Adaptive Runtime Binary Optimization Framework for Multithreaded Applications”, International Conference on Parallel Processing (ICPP), 2007, pp: 25-33.

[2] L. Baraz, T. Devor, O. Etzion, et.al, “IA-32 Execution Layer: a two-phase dynamic translator designed to support IA-32 applications on Itanium-based systems.” In 36th International Symposium on Microarchitecture, 2003, pp: 191-201.

[3] HP ARIES Dynamic Binary Translator, http://h21007.www2.hp.com.

[4] LUK, C.-K., COHN, R., MUTH, R., et.al, “Pin: building customized program analysis tools with dynamic instrumentation”. In PLDI ’05 (New York, NY, USA, 2005), pp. 190–200.

[5] V. Bala, E. Duesterwald, and S. Banerjia, “Dynamo:

A transparent runtime optimization system". In Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '00), June 2000.

[6] Nethercote, N., Seward, J. Valgrind: a framework for heavyweight dynamic binary instrumentation. In PLDI '07 (New York, NY, USA, 2007), pp. 89–100.

[7] Swaroop Sridhar, Jonathan S. Shapiro, Prashanth P. Bungale, et.al, "HDTrans: An Open Source, Low-Level Dynamic Instrumentation System", In VEE '06: Proceedings of the 2nd international conference on Virtual execution environments (2006), pp. 175-185.

[8] B. Cmelik and D. Keppel, "Shade: A fast instruction-set simulator for execution profiling", In Proceedings of the 1994 ACM SIGMETRICS Conference on the Measurement and Modeling of Computer Systems (1994), pp. 128–137.

[9] Fabrice Bellard. "QEMU: a Fast and Portable Dynamic Translator", Proceedings of the USENIX Annual Technical Conference, 2005, pp.41-46.

[10] K Ebcioglu, E R Altman, "DAISY: Dynamic Compilation for 100% Architectural Compatibility", 24th Annual International Symposium on Computer Architecture (ISCA'97), pp: 26-37.

[11] J. Lu, H. Chen, P.-C. Yew, et.al, "Design and implementation of a lightweight dynamic optimization system," The Journal of Instruction-Level Parallelism, vol. 6, 2004.

[12] Youfeng Wu, Mauricio Breternitz, Justin Quek, Orna Etzion, Jesse Fang. "The Accuracy of Initial Prediction in Two-Phase Dynamic Binary Translators". Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization. pp:227-238.

[13] A. Chernoff and R. Hookway, "DIGITAL FX!32 - Running 32-Bit x86 Applications on Alpha NT," USENIX Association. Berkeley CA: Proceedings of the USENIX Windows NT Workshop, August 1997.

[14] Chaohao Xu, Jianhui Li, Tao Bao, Yun Wang, "Metadata driven memory optimizations in dynamic binary translator", Proceedings of the 3rd international conference on Virtual execution environments. San Diego, California, USA. Pages: 148 – 157, 2007.

[15] GNU Compiler Collection Internals.

[16] K. Hazelwood and M. D. Smith. "Managing bounded code cache in dynamic binary optimization systems", Transaction on Code Generation and Optimization, 3:263-294, 2006.

[17] K. Hazelwood and J. E. Smith, "Exploring Code Cache Eviction Granularities in Dynamic Optimization Systems", Proceeding of the International Symposium on Code Generation and Optimization, pages 89-99, 2004.

[18] A. Guha, K. Hazelwood and Mary Lou Soffa, "Reducing Exit Stub Memory Consumption in Code Caches", High Performance Embedded Architecture and Compilers, Second International Conference, pages 87-101, 2007.

[19] D. L. Bruening and S. Amarasinghe, "Maintaining Consistency and Bounding Capacity of Software Code Caches", Proceedings of the International Symposium on Code Generation and Optimization, pages74-85, 2005.

[20] D. L. Bruening and V. Kiriansky, "Process-Shared and Persistent Code Caches", Proceedings of the 4th International Conference on Virtual Execution Environment, pages 61-70, 2008.

[21] D. L. Bruening, "Efficient, Transparent, and Comprehensive Runtime Code Manipulation", Ph.D thesis, Massachusetts Institute of Technology (2004).

[22] Apala Guba, Kim Hazewood and Mary Lou Soffa, "DBT Path Selection for Holistic Memory Efficiency and Perfomance", VEE'10 March 17-19, 2010, Pittsburgh, Pennsylvania, USA.

[23] David Hiniker, Kim Hazelwood, Michael D. Smith, "Harvard UniversityImproving Region Selection in Dynamic Optimization Systems", Proceedings of the 38th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'05).

[24] Duane Merrill, Kim Hazelwood, "Trace Fragment Selection within Method-based JVMs", VEE'08, March 5-7, 2008, Seattle, Washington, USA.

[25] D. Bruening, T. Garnett, S. Amarasinghe. "An infrastructure for adaptive dynamic optimization", In International Symposium on Code Generation and Optimization, pages 265-275, San Francisco, California, 2003.

[26] Andreas Krall, "Improving Semi-static Branch Prediction by Code Replication", SIGPLAN 94-6/94 Orlando, Florida USA.

[27] Yunchao He, Chen Kai, Jinghui Gu, et.al, "A New Approach to Reorganize Code Layout of Software Cache in Dynamic Binary Translator", (PAAP2010). Accepted.

[28] Haipeng Deng, Kai Chen, Bo Liu, et.al, "Efficient Online Trace Building Using Code Replication", GCC 2010, Accepted.

[29] Yindong Yang, Haibing Guan, Erzhou Zhu, Hongbo Yang, Bo Liu, CrossBit: A Multi-Sources and Multi-Targets DBT, CLOUD COMPUTING 2010, November 21-26, 2010 - Lisbon, Portugal, accepted.

[30] Jinghui Gu,Chao Xu,Ling Lin,Juyu Zheng,Kai Chen,Haibing Guan, The Implementation of Static-integrated Optimization Framework for Dynamic Binary Translation, ITCS2009, Kiev, Ukraine, 25-26 July, 2009.

[31] Huihui Shi, Yi Wang, Haibing Guan, Alei Liang,"An Intermediate Level Optimization Framework for DBT", ACM SIGPLAN notice, vol 42(5),2007.5:3~9.

Haibing Guan received his Ph.D. degree in computer science from the TongJi University (China), in 1999. He is currently a professor with the Faculty of Computer Science, Shanghai Jiao Tong University (Shanghai, China). His current research interests include, but are not limited to, computer architecture, compiling, virtualization and hardware/software co-design.

Erzhou Zhu is currently a Ph.D student at Shanghai Jiao Tong University, China. He received the M.S. degree and B.S. degree in Computer Science and Technology in Anhui University, Anhui, China, in 2004 and 2008 respectively. His research interests include virtual machine, binary translation and computer architecture.

Kai Chen received his Ph.D. degree in computer science from Shanghai Jiaotnog University (China). He is currently a lecturer with the Faculty of information security engineering, Shanghai Jiaotong University (Shanghai, China). His current research interests include, but are not limited to, computer architecture, network virtualization, virtual machines.

Ruhui Ma is currently a Ph.D student at Shanghai Jiao Tong University, China. His research interests include virtual machine, binary translation and computer architecture.

Yunchao He, master candidate. Got bachelor degree from School of Computer Science, Wuhan University of Science and Technology in 2008. Now I am study in Embedded System Lab, School of Software, Shanghai Jiao Tong University in China. My research area is dynamic binary translation and I/O virtualization on KVM.

Haipeng Deng is currently a Master Degree Candidate student at Shanghai Jiao Tong University, China. His main research interest is binary translation.

Hongbo Yang is currently a Ph.D. student at Shanghai Jiao Tong University, China. He received the M.S. degree in 1995 and received his B.S. degree in 1998 at Institute of Airforce Meteorologyity, China. His main research interests are in virtual machines, computer architecture and compiling.

An Efficient Hybrid Clustering-PSO Algorithm for Anomaly Intrusion Detection

Hongying Zheng
College of Computer Science and Engineering, Chongqing University, Chongqing 400044, China
zhenghongy@cqu.edu.cn

Meiju Hou, Yu Wang
College of Computer Science and Engineering, Chongqing University, Chongqing 400044, China
meijuhou0119@126.com，chuang_bei@sohu.com

***Abstract*—Generally speaking, in anomaly intrusion detection, modeling the normal behavior of activities performed by a user or a program is an important issue. Currently most machine-learning algorithms which are widely used to establish user's normal behaviors need labeled data for training first, so they are computational expensive and sometimes misled by artificial data. This study proposes a PSO-based optimized clustering method IDCPSO for modeling the normal patterns of a user's activities which combines an unsupervised clustering algorithm with the PSO technique, PSO algorithm is used to optimize the clustering results and obtain the optimal detection result. IDCPSO needs unlabeled data for training and automatically establishes clusters so as to detect intruders by labeling normal and abnormal groups. The famous KDD Cup 1999 dataset is used to evaluate the proposed system. In addition, we compare the performance of PSO optimization process with GA.**

***Index Terms*—PSO, Unsupervised Clustering, Anomaly Intrusion Detection, Optimization**

I. INTRODUCTION

Currently, Due to the advance of computer and communication technology and the reliance on Internet and world wide connectivity, damages caused by unexpected intrusions and crimes related to computer systems have been increased rapidly; A computer system should provide confidentiality, integrity and availability against denial of service; therefore, it is very important that the security mechanisms of a system are designed so as to prevent unauthorized access to system resources and data. Firewalls are hardware or software systems placed in between two or more computer networks to stop the committed attacks by isolating these networks using the rules and policies determined for them. However, it is very clear that firewalls are not enough to secure a network completely because the attacks committed from outside of the network are stopped whereas inside attacks are not. This is the situation where intrusions detection systems (IDSs) are in charge. IDSs are used in order to stop attacks, recover from them with the minimum loss or analyze the security problems so that they are not repeated, the performed tasks of an IDS include [1]: (1) Monitoring and analyzing user and system activities;(2) Audit of system structure and fault;(3) Recognition activity model mapping known attacks and alert;(4) Statistic analysis of abnormal behavior model;(5) Evaluating the integrity of systems and data files. Intrusion detection systems assume that they can detect an intruder by examining such parameters as network traffic, CPU and I/O utilization, user location, and file activity for signs of an attack.

In order to detect intruders, a number of detection methods have been proposed, these methods are mostly based on Denning's intrusion detection model. In this model, audit records, network packets and any other observable activity service as the basis for detecting abnormalities in the system. These techniques can be categorized into anomaly detection and misuse detection. Anomaly detection systems flag observed activities that deviate significantly from the established normal usage patterns as anomalies (i.e. possible intrusion). Therefore, the main issues in anomaly detection systems thus become the selection of threshold levels so that neither of the false negative rate and the false positive rate is unreasonably magnified. The main advantage of anomaly detection approaches is the ability to detect novel attacks or unknown attacks against systems, variants of known attacks, and deviation of normal usage of programs. And the disadvantage of anomaly detection approaches is that if the attack fits the established profile of the user, it is often difficult to detect it; another drawback is that a malicious user can train the anomaly detection system to learn the attack's malicious behaviors as normal by changing the profile slowly over time. Finally, anomaly intrusion detection methods always result in high false positive rate. While misuse detection systems, use patterns of well-known attacks or weak spots of the system to match and identify known intrusion patterns or signatures. The main issues in misuse detection systems are how to write signatures or patterns that encompass all possible variations of the attacks. The main advantage of misuse detection approaches is that they have high detection rare. Whereas, Novel attacks or unknown attacks or even

doi:10.4304/jsw.6.12.2350-2360

variants of common attacks often go undetected.

Unsupervised anomaly detection methods have been addressed recently, these methods take a set of unlabeled data as input and attempt to find intrusion buried within the data. To detect a new attack, they do not need any prior knowledge about training data and new attacks. Clustering is the unsupervised classification of input items into groups (clusters) without any prior knowledge. It is promising to detect unknown attacks in intrusion detection automatically. Furthermore, the clusters data can be analyzed for more information, i.e. the signature of new attacks. Most unsupervised anomaly detection methods are based on two basic assumptions about data. First, the number of normal instances vastly outnumbers that of anomalies. Second, data instances of the same classification (type of attack or normal) should be close to each other in the feature space under some reasonable metrics, and instances of different classifications are far apart. For example, Portnoy and Eskin presented a clustering-based anomaly detection approach in Ref. [2] which used the labeled clusters to classify network data according to the label of the nearest cluster. To overcome the drawback of a finite audit data set indicated the static activity of a user, authors in Ref. [3] proposed an anomaly intrusion detection method that continuously modeled the normal behavior of a user over the audit data stream, clusters of feature values corresponding to activities observed thus far in an audit data stream were identified by a statistical grid-based clustering algorithm for a data stream. In some cases, the clustering method is used to provide other algorithms with higher-qualified training items before training. For example, authors in Ref. [4] proposed an SVM-based intrusion detection system, which combined a hierarchical clustering algorithm and the SVM technique. The hierarchical clustering algorithm provided the SVM with fewer, abstracted, and higher-qualified training instances that were derived from the KDD Cup 1999 training set. It was able to greatly shorten the training time, but also improve the performance of resultant SVM. In Ref. [5], a new RFID intrusion detection method that was based on fuzzy c-Means clustering was proposed, the new RFID intrusion detection method can enhance the security and speed up the intrusion detection of RFID systems. In Ref. [6], the author proposed a new algorithm based on time stamped hierarchical clustering for intrusion detection. It incrementally clustered incoming data objects of normal behavior, and produced a precise model. Using each clusters time stamp, the algorithm can dynamically remove some expired clusters from the model, and also can produce some new cluster, which made clustering method more suitable for real network environment, the algorithm was less sensitive to noise data objects, and had low computer resource consumptions. In Ref. [7], the author investigated a hybrid clustering based filtering approach for high dimensional data clustering in detecting anomaly based network intrusions. A hierarchical conceptual clustering algorithm (COBWEB) had been used for data filtering and Farthest First Traversal (FFT) clustering technique for classification of rare attacks, the proposed approach was quite effective in comparison to their individual counterparts in detecting network intrusions, especially that come under U2R and R2L rare attacks category.

Although many kinds of clustering methods, such as FCM, K-MEANS, are widely used in intrusion detection, few clustering algorithms guarantee a global optimal solution. Therefore, to find global optimal clusters instead of local optimal results, this article presents a new approach for network anomaly intrusion detection called IDCPSO (Clustering and PSO). In the proposed method, the authors model the normal behaviors of a user or a program by using clustering the training data set and the optimal clustering results are obtained by means of PSO algorithm. PSO has many advantages over other evolutionary computation techniques (for example, genetic algorithm (GA)) such as simple implementation, faster convergence rate and fewer parameters to adjust. PSO is presented so as to combine it with clustering techniques for finding the minimum of the fitness function, producing a good result and enhancing the detection rate of intrusion. IDCPSO can combine the advantages of both PSO with clustering. Results are also compared with genetic algorithm. The procedure of intrusion detection is composed of the three parts, that is, (1)modeling the normal behaviors of a user or a program by creating clusters from unlabeled training datasets; (2) labeling clusters as ‘normal’ or ‘anomalous’; (3) using the labeled clusters to classify network data.

The remainder of this paper is organized as follows. Section 2 discusses the related work of anomaly intrusion detection based on supervised and unsupervised learning. Section 3 describes the proposed method of IDCPSO in detail. Section 4 shows the experiment results on KDD cup. Section 5 gives some conclusions.

II. THE RELATED WORK

In general, depending on whether the class labels are provided for learning, anomaly intrusion detection algorithms can be classified as either supervised or unsupervised. The supervised algorithms mostly exhibit excellent classification accuracy on the data with known attacks, but the accuracy of supervised algorithms deteriorates significantly if unknown attacks are present in the test data. While the unsupervised algorithms exhibit no significant difference in performance between known and unknown attacks, in other words, the performance of unsupervised learning is not affected by unknown attacks [8].

A. Supervised Anomaly Detection

Many supervise anomaly detection methods were proposed, for example k-Nearest Neighbor, SVM, MLP, and Decision Trees and so on, they are described in detail as follows.

The k-Nearest Neighbor is a classical algorithm that

finds k examples in training data that are closest to the test example and assigns the most frequent label among these examples to the new example. The only free parameter is the size k of the neighborhood. In Ref. [9], the task of the nearest neighbor approach was to remove noisy data and to build the set of original clusters. The aim of clustering by nearest neighbor algorithm was to reduce the size of data sets to a moderate one suitable for genetic algorithms at the second stage and to reduce the computing time as much as possible. In Ref. [10], the author used kNN classifier which was also a popular method in text categorization. The kNN classifier was used to classify program behavior as normal or intrusive. Program behavior was represented by frequencies of system calls. Each system call was treated as a word and the collection of system calls over each program execution as a document. This method seemed to offer some computational advantages over those that seek to characterize program behavior with short sequences of system calls and generated individual program profiles. In Ref. [11], the authors proposed a hybrid learning model based on the triangle area based nearest neighbors (TANN) in order to detect attacks more effectively. In TANN, the k-means clustering was firstly used to obtain class centers corresponding to the attack classes, respectively. Then, the triangle area by two class centers with one data from the given data set was calculated and formed a new feature signature of the data. Finally, the k-NN classifier was used to classify similar attacks based on the new feature represented by triangle areas.

Some supervised algorithms generalize equally well to the data with unknown attacks, for example SVM, which can be attributed to the fact that the free parameters of this algorithm are motivated by learning-theoretic arguments aimed at maintaining an ability to generalize to unseen data. In Ref. [12], the author applied the ANN and SVM algorithms to employ a frequency-based encoding method Instead of using a conventional sequence-based encoding scheme for intrusion detection. SVM is a powerful tool for classification problems, but still has some drawbacks. The first problem is that SVM is sensitive to outliers or noises. Second, SVM is designed for the two-class problem, so it has to be extended for multiclass problems. Some methods to improve the performance of SVM were proposed. In Ref. [13], the author projected input data into a high dimensional space by using the discriminant vectors extracted by KFDA. Then they constructed the optimal decision tree for multiclass SVM, based on the results of fuzzy clustering on the projected data. In Ref. [14], an improved incremental SVM algorithm, named RS-ISVM, was developed. To reduce the noise generated by feature differences, the author proposed a modified kernel function U-RBF, with the mean and mean square difference values of feature attributes embedded in kernel function RBF. Moreover, in order to shorten the training time, a concentric circle method was suggested to be used in selecting samples to form the reserved set.

Training of a multi-layer perception involves optimizing the weights for the activation function of neurons organized in network architecture. The free parameter is the number of hidden neurons. In Ref. [15], a hybrid MLP/CNN neural network was constructed in order to improve the detection rate of time-delayed attacks, the proposed approach can detect time-delayed attacks efficiently with chaotic neuron and this approach also exhibited a lower false alarm rate when detects novel attacks.

Decision trees build classification models based on recursive partitioning of data. Typically, a decision tree algorithm begins with the entire set of data, splits the data into two or more subsets based on the values of one or more attributes, and then repeatedly splits each subset into finer subsets until the size of each subset reaches an appropriate level. The entire modeling process can be represented in a tree structure, and the model generated can be summarized as a set of "if-then" rules. Decision trees are easy to interpret, computationally inexpensive, and capable of coping with noisy data. Therefore, the techniques have been widely used in intrusion detection. In Ref. [16], SURPASS was proposed which was highly effective in handling large data. SURPASS incorporates linear discriminants into decision trees' recursive partitioning process. In SURPASS, the information required to build a decision tree was summarized into a set of sufficient statistics. By reading a subset of the data from storage space to main memory one at a time, the data size that can be handled by this algorithm was independent of memory size.

B. Unsupervised Anomaly Detection

While in unsupervised learning, the data are not labeled. From the perspective of machine learning, the searching for clusters is unsupervised learning. To perform clustering is to try to discover the inner nature of the data structure as a whole, and to divide the data into groups of similarity. For example, in Ref. [17], The RT-UNNID system was introduced to be capable of intelligent real-time intrusion detection using unsupervised neural networks. Unsupervised neural nets can improve their analysis of new data over time without retraining. The aim of using unsupervised neural nets was to detect known and new attacks in network traffic. Some researchers use SOMs [18, 19] to learn patterns of normal system activities in anomaly detection tasks and employ a three-layer SOM to detect anomalous user behavior and anomalous network traffic.

There are generally three types of clustering algorithms that is Partition-based clustering, Hierarchical clustering and Density based clustering.

Partition-based clustering divides data sets into non-overlapping clusters. Given a predefined number of clusters, find the optimal partition for each point. The k-means algorithm is a well-known example of this kind of clustering methods. In Ref. [20], based on developed HSMM, an algorithm of anomaly detection was presented, which computed the distance between the

processes monitored by intrusion detection system and the perfect normal processes. To improve accuracy, the segmental K-means algorithm was applied as training algorithm for the hidden semi-Markov model. In Ref. [25], the author proposed a novel algorithm (IDCPSO) for modeling the normal behavior of user activities. There are two stages in the implementation of the algorithm. The first stage is to cluster network data and the second is the optimization process. Compared to genetic algorithms, this paper demonstrated higher DR and lower FPR and FNR.

While hierarchical clustering builds a cluster hierarchy. The hierarchy is a tree of clusters. Every node in the tree contains child clusters while sibling clusters share a common parent node. Typical hierarchical clustering algorithms are BIRCH and CURE. In BIRCH, a CF (Clustering Feature) tree which was used to summarize cluster representations was generated dynamically. After the CF tree was built, any clustering algorithm such as a typical partitioning algorithm was then used. For example, in Ref. [4], BIRCH clustering algorithm was first used to produce a reduced and high quality dataset from the original KDD Cup 1999 dataset before SVM training. The proposed system constructed four CF trees for DoS, U2R, R2L, and Probe attacks, respectively, and one CF tree for normal packets. The proposed system could reach high detection accuracy with a low false positive rate.

Unlike other methods, the density based clustering method regards a cluster as a region in a data space with the proper density of data objects. Typical density-based clustering algorithms are DBSCAN and CLIQUE. In Ref. [21], the author proposed an anomaly detection method, which utilized a density-based clustering algorithm DBSCAN for modeling the normal behavior of a user's activities in a host. The common knowledge of activities in the transactions of a user was represented by the occurrence frequency of similar activities by the unit of a transaction as well as the repetitive ratio of similar activities in each transaction. In Ref. [22], a new density-based and grid-based clustering algorithm that is suitable for unsupervised anomaly detection was presented. The system can be trained with unlabelled data and was capable of detecting previously unseen attacks.

C. Hybrid Anomaly Detection

In related work, a hybrid machine learning model based on combining the unsupervised and supervised classification techniques was proposed. One clustering technique is used as the first component for "pre-classification" and one supervised classification technique as the second component for the final classification task. For example, Latifur Khan et al. [23] presented a new approach that is combination of Support Vector Machines and Dynamically Growing Self-Organizing Tree (DGSOT) algorithm. SVM was used for classification and hierarchical clustering for analysis. Clustering analysis helped find the boundary points, which were the most qualified data points to train SVM, between two classes. The method started with an initial training set and expanded it gradually using the clustering structure produced by the DGSOT algorithm. In Ref. [24], Cheng proposed a multiple-level hybrid classifier which combines the supervised tree classifiers and unsupervised Bayesian clustering to detect intrusions, this new approach had high detection and low false alarm rates.

III. THE IDCPSO ALGORITHM

The IDCPSO algorithm consists of two stages. They are stated as follows:

1) The stage of clustering: To establish the set of original clusters using the clustering method by grouping very similar instances into a cluster and filter noisy data objects based on some similarity or dissimilarity metrics.

2) The stage of PSO optimization: To optimize original clusters by PSO algorithms and obtain the near optimal result, then label the cluster including most activities as the normal according to above assumptions.

A. Unsupervised Clustering

Clustering analysis is a method for grouping and classifying objects without category labels. Let the set of n points $\{ c_1, c_2, \ldots, c_n \}$ be represented by the data set Q and the number of clusters be represented by K, In Ref. [26], clustering is described as an assignment problem. Each cluster has a unique cluster label in $(1,...,K)$; the vector c assigns a cluster label $c_i \in (1,...,K)$ to the i-th point. Fig.1 shows an assignment instance. the number of data item is equal to 5, namely |Q|=5, K=2 and the assignment vector c=(2,2,1,2,1). The vector c shows that the data set is divided into two clusters, the first cluster is composed of data items 3, 5 and the other cluster is data items 1, 2, 4.

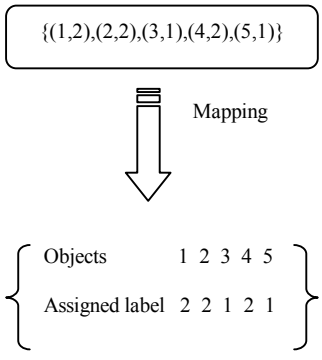


Figure 1. The assignment instance

For an assignment, we can evaluate the clustering quality by clustering criterion. For example, squared error based on cluster mean in (1) which is widely used in k-means. m_k denotes the vector of cluster center of the k-th cluster and $\|\bullet\|$ is Euclidean distance. Besides this, other clustering criterions are taken into account.

Such as (2) and (3) which define the total of inter-cluster distance and intra-cluster distance separately.

$$J = \sum_{k=1}^{K} \sum_{c_i=k} \left\| c_i - m_k \right\| \quad (1)$$

$$J = \sum_{k=1}^{K} \sum_{\substack{c_i=k \\ c_j=k}} \left\| c_i - c_j \right\| \quad (2)$$

$$J = \sum_{k=1}^{K} \sum_{\substack{c_i=k \\ c_j \neq k}} \left\| c_i - c_j \right\| \quad (3)$$

For a given data set, $\left(c : \forall i \in (1,...,N), c_i \in (1,...,K)\right)$ is the set of all feasible clustering, we should find the optimal solution of c about clustering criterion to enhance clustering quality. Therefore, clustering is converted into the optimizing problem described as the following (4) and (5).

$$\text{Minimize } J(c) \quad (4)$$

$$\text{Subject to } c \in C \quad (5)$$

B. The Stage of PSO Optimization

After above steps, the set of initial clusters can be obtained. This stage is a PSO optimization process at which initial clusters are used to set up the initial particle. The PSO optimizes a fitness function by iteratively improving a swarm of solution vectors, called particles, based on special management of memory. Each particle is modified by referring to the memory of individual and swarm's best information. Due to the collective intelligence of these particles, the swarm is able to repeatedly improve its best observed solution and converges to an optimum. This will be described in the following.

1) Framework of PSO

The PSO method was developed by Kennedy and Eberhart [27, 28] which has been successfully applied to many science and practical fields [29, 30, 31, 32, 33, 34]. PSO is a sociologically inspired population based optimization algorithm. Each particle is an individual, and the swarm is composed of particles. In PSO, the solution space of the problem is formulated as a search space. Each position in the search space is a correlated solution of the problem. Particles cooperate to find the best position (best solution) in the search space (solution space). Each particle moves according to its velocity. During iteration, the particle movement is computed as follows:

$$x_i(t+1) \leftarrow x_i(t) + v_i(t) \quad (6)$$

$$v_i(t+1) \leftarrow w v_i(t) + c_1 r_1 (pbest_i(t) - x_i(t)) + c_2 r_2 (gbest(t) - x_i(t)) \quad (7)$$

In (6), (7), $x_i(t)$ is the position of particle i at time t, $v_i(t)$ is the velocity of particle i at time t, $pbest_i(t)$ is the best position found by particle i itself so far, $gbest(t)$ is the best position found by the whole swarm so far, w is an inertia weight scaling the previous time step velocity, c_1 and c_2 are two acceleration coefficients that scale the influence of the best personal position of the particle ($pbest_i(t)$) and the best global position ($gbest(t)$), which are popularly chosen to be $c_1 = c_2 = 2$, r1 and r2 are random variables between 0 and 1. The process of PSO is shown as Fig.2.

```
Initialize a population of particles with random positions and velocities in
the search space.
While (termination conditions are not met)
{
  For each particle do i
  Update the position of particle i according to (6).
  Update the velocity of particle i according to (7).
    Map the position of particle i in the solution space and evaluate its
      fitness value according to the fitness function.
  Update pbest_i(t) and gbest(t) if necessary.
  End for
}
```

Figure 2. The process of the PSO algorithm

2) Particle representation and initial swarm generation

We let each particle represent a decision for cluster assignment using a vector of N elements and each element is binary value. N is the number of data items in data set Q, namely |Q|=N. Fig.3 shows an illustrative example for the ith particle which corresponds to a cluster assignment that the data items 1,2,4 are belong to a cluster and the items 3,5 are the other cluster .

The PSO randomly generates an initial swarm of M particles, where M is the swarm size. These particle vectors will be iteratively modified based on collective experiences in order to improve their solution quality.

	1	2	3	4	5
The ith particle:	1	1	0	1	0

Figure 3. An example for the ith particle representation

3) Fitness function

The fitness function represents the goodness degree of a solution. In IDCPSO, we take into account not only intra-cluster distance but also inter-cluster distance. The intra-cluster distance indicates the degree of apart from; the larger the value, the farther distance the two clusters. Let $d(X,Y)$ be the intra-cluster distance. While the inter-cluster distance shows the degree of similarity. The less the distance, the more similarity the two data items in the same cluster. Let x_i, y_i be the input data items, m_i is the cluster center equaling to the mean of data items which belong to the same cluster. Therefore, the

fitness function of the particle vector can finally be defined as (8).

$$J' = \left(J_X + J_Y\right) \Big/ d_{(X,Y)} \tag{8}$$

$$J_X = \sum_{i=1}^{N}\sum_{j=1}^{N}\left\|x_i, x_j\right\| \tag{9}$$

$$d_{(X,Y)} = \left\|m_X, m_Y\right\| \tag{10}$$

$$m_X = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{11}$$

4) Particle vector modification

In IDCPSO, since the particle position is a binary string, the particle position vector modification is completed using (12), where $Sig(x) = 1/(1+\exp(-x))$. The parameter r_3 is uniformly distributed random numbers in [0, 1]. The particle flies through potential solutions toward pbesti and gbest in a navigated way while still exploring new areas by the stochastic mechanism to escape from local optimum [29].

$$x_{id}(t+1) = \begin{cases} 1, r_3 < Sig(v_{id}(t+1)) \\ 0, r_3 \geq Sig(v_{id}(t+1)) \end{cases} \tag{12}$$

5) Stop criterion

The IDCPSO will be terminated if the max number of iterations is met or the fitness function does not change any more in a given number (the algorithm converges to an optimum).

C. Labeling Clusters

Let an assumption that normal data items constituting an overwhelmingly large portion of the training dataset be satisfied. We therefore label some percentage N of the clusters containing the largest number of data items associated with them as 'normal', the rest of the clusters are labeled as 'anomalous'. Besides this, there is another problem that should been taken into consideration, that is, there may be many different kinds of normal network activity, this, in turn, might produce a large number of such 'normal' clusters which will have a relatively small number of data items. Therefore, for labeling a cluster, we should calculate the number of data items as well as consider the distance from other clusters. If the number of data items in a cluster is lowest and the distance from the other clusters is largest, we labeled it ‘anomalous’.

D. Anomaly Detection Method

Once the clusters are created from a training data set, the system is ready to perform detection of intrusions. Given a data item d, we should standardize d to d’ and find a cluster which is closest to d’ under the Euclidean distance, i.e. a cluster c in the cluster set C, such that for all C-c, dist (d’, c) ≤ dist (d’, C-c) .Assign type of cluster c (either normal or anomalous) to data item d.

IV. EXPERIMENTAL RESULTS

In this section, we describe the experiments conducted to evaluate the proposed system. The proposed system is tested on a Celeron processor 1.5 GHz with 512 RAM running Windows XP and coded by matlab 6.5.

A. Data Source

The KDD Cup99 dataset [35] is derived in 1999 from the DARPA98 network traffic dataset by assembling individual TCP packets into TCP connections. It is the benchmark dataset used in the International Knowledge Discovery and Data Mining Tools Competition, and also the most popular dataset that has ever been used in the intrusion detection field. Each TCP connection has 41 features with a label which specifies the status of a connection as either being normal, or a specific attack type. There are 38 numeric features and 3 symbolic features falling into the following four categories:

1) Basic features: 9 basic features are used to describe each individual TCP connection.

2) Content features: 13 domain knowledge related features are used to indicate suspicious behavior having no sequential patterns in the network traffic.

3) Time-based traffic features: 9 features are used to summarize the connections in the past 2 s that have the same destination host or the same service as the current connection.

4) Host-based traffic features: 10 features are constructed using a window of 100 connections to the same host instead of a time window.

Table I shows the basic features of KDD cup99. Because of the limitation of searching space and time, dimensionality should be reduced. finally, only 7 features are chosen. They are src_bytes, dst_bytes, count, srv_count, dst_host_count, dst_host_srv_count, dst_host_same_src_port_rate.

B. Data Preparing

The dataset contains about five million connection records as training data and about two million connection records as test data. Attacks fall into four categories: (1) Denial of Service (DoS): making some computing or memory resources too busy to accept legitimate users access these resources. (2) Probe (PRB): host and port scans to gather information or find known vulnerabilities. (3) Remote to Local (R2L): unauthorized access from a remote machine in order to exploit machine’s vulnerabilities. (4) User to Root (U2R): unauthorized access to local super user (root) privileges using system’s susceptibility.

In order to reduce the size of the dataset, we randomly select 952 records which satisfy the above assumption. Table II shows detailed information about the number of all records. It is important to note that the test data includes specific attack types not present in the training

data. This makes the intrusion detection task more realistic.

TABLE I
THE BASIC FEATURE NAMES

Feature name	Description	Type
Duration	length (number of seconds) of the connection	Count
Protocol_type	type of the protocol, e.g. tcp, udp, etc.	String
Service	network service on the destination, e.g., http, telnet	String
Src_bytes	number of data bytes from source to destination	Count
Dst_bytes	number of data bytes from destination to source	Count
Flag	normal or error status of the connection	String
Land	1 if connection is from/to the same host/port; 0 otherwise	Boolean
Wrong_fragment	number of ``wrong'' fragments	Count
Urgent	number of urgent packets	Count

TABLE II
NUMBER AND DISTRIBUTION OF TRAINING AND TESING DATASET

Connection type	Training dataset	Testing dataset
Normal	633(66.49%)	2000(44.56%)
DoS	96(10.08%)	350(7.8%)
PRB	65(6.83%)	560(12.48%)
R2L	73(7.67%)	678(15.11%)
U2R	85(8.93)	900(20.05%)

TABLE III.
DESCRIPTIVE STATISTICS OF THE 952CASES OF THE DATASET IN KDD CUP99

Feature name	Type	m_j (1.0e+003)	s_j (1.0e+003)
src_bytes	Continuous	0.3808	0.1601
dst_bytes	Continuous	2.0245	7.8162
count	Continuous	0.2782	0.2467
srv_count	Continuous	0.2733	0.2507
dst_host_count	Continuous	0.1935	0.0959
dst_host_srv_count	Continuous	0.2388	0.0588
dst_host_same_ src_port_rate	Continuous	0.0006	0.0005

C. Data Normalization Processing

We encounter a problem when processing instances whose different features are on different scales. This will cause bias toward some features over other ones. To solve this problem, these raw data sets need to be normalized. 41 features can be divided into 4 categories (‘Boolean’, ’String’, ’Count’, ’Rate’). Category ‘Boolean’ is 0 for ‘no’ and 1 for ‘yes’; category ‘Count’ is normalized according to (13); category ‘Rate’ remains unchanged. Category ‘String’ is mainly used to analyze features of clusters. m_j is the average feature instance of j-th feature and s_j is the standard deviation. Let the input data set be $I = \{I_{ij} \mid i = 1,...,N, j = 1,...,P\}$, where N is the number of total sessions and P is the number of features. Table III gives statistic results about m_j and s_j.

$$I'_{ij} = \frac{I_{ij} - m_j}{s_j} \tag{13}$$

$$m_j = \frac{1}{N}\sum_{i=1}^{N} I_{ij} \tag{14}$$

$$s_j = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(I_{ij} - m_j)^2} \tag{15}$$

D. Evaluation Criteria

For evaluating the IDS outputs (in the test phase), an IDS Evaluator component is added to IDCPSO. This component, by comparing the output of IDS and expected output of the system (which is determined for a test data set in a separate file), calculates the following evaluation metrics:

1) DR: true detection rate (only separating normal traffic from attacks);

2) FPR: false positive detection rate (mis-detecting attacks);

3) FNR: false negative detection rate (failing to detect attacks when they are occurred).

False negatives are dangerous problems, and are far more serious than the problem of false positives.

$$DR = \frac{the\ \ number\ \ of\ \ attacks\ \det ected}{the\ \ number\ \ of\ \ attacks} \tag{16}$$

$$FPR = \frac{the\ \ number\ \ of\ \ false\ \ positive}{false\ \ positive + true\ \ positive} \tag{17}$$

$$FNR = \frac{the\ \ number\ \ of\ \ false\ \ negative}{false\ \ negative + true\ \ negative} \tag{18}$$

E. Parameters Setup

In order to achieve the best performance of IDCPSO, we implement the experiments with different kinds of parameters setup. From the Fig.4, 5, we find that the convergence process of fitness function with c1=c2=2 is better than that of other value (c1=c2=4), and we can get the best value of fitness function when we set c1=c2 rather than c1≠c2. Determination of particle population size is still very important, if the value is set too small, we do not get the optimal value; conversely, if the value is set too large, the iteration time will be too long. Fig. 6 describes the relation between the population size and the average fitness value. Fig.7 shows the convergence process with different maximum velocity. We can obtain the best convergence when we set the maximum velocity equal to 10(vmax=10). Fig.8 explains the settings on the inertia weight value problem. The most reasonable setting of inertia weight value is w=1.0.

Therefore, the final parameters setup is showed in table IV.

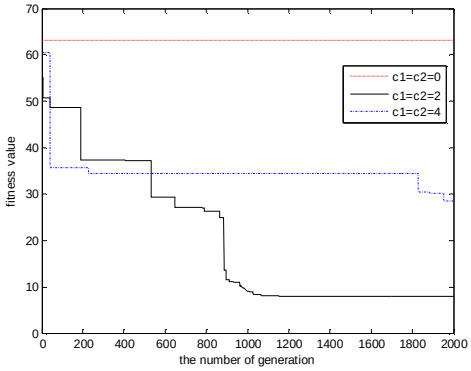


Figure 4. The convergence process when $c_1 = c_2$

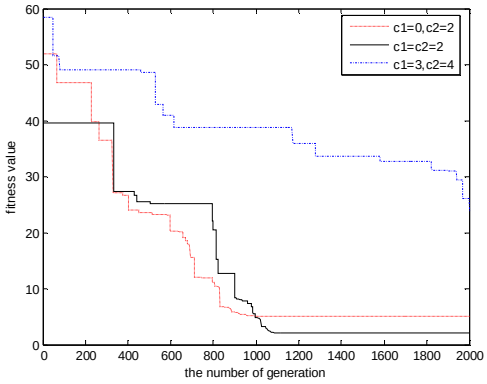


Figure 5.The convergence process using different acceleration coefficients value

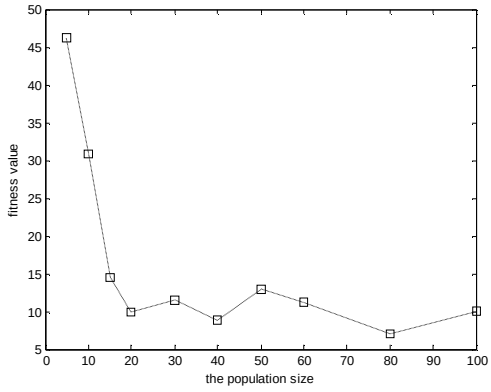


Figure 6. The relation between the population size and the average fitness value

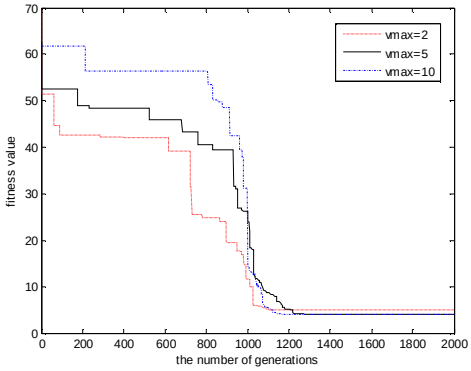


Figure 7. The convergence process with different vmax

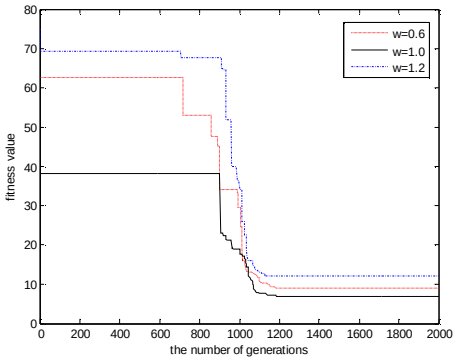


Figure 8. The convergence process with different inertia weight values

TABLE IV.
THE IDCPSO ALGORITHM PARAMITERS SET

Parameter	Value
$c_1 = c_2$	2
w	1
$v_{\max}$	10
Particles	20
Iterations	1500

F. Performance Comparison

To evaluate and compare the performance of the proposed algorithm, it is compared with Genetic Algorithm (GA), because GA is also population-based evolutionary computation algorithm which is widely used in many areas [36, 37, 38]. The comparison result about the convergence process of fitness function is given in Fig.9. It is obvious that the convergence speed of IDCPSO is faster and the number of iterations is less than that of GA. The detection performance is shown in Table V. According to the results, IDCPSO has higher DR, lower FPR and FNR than that of GA. We can obtain a better value about fitness function in much less generations, therefore, detection results is better than that of GA. In fig.10, 11, as the number of particles decreases, both the detection results and the convergence value decrease dramatically in GA; especially it is obvious when the number of particles is equal to 5. As for IDCPSO, we can find that not only the detection rate but also the convergence result are unchangeable. Therefore, the performance of GA is more sensitive to the number of particles than that of IDCPSO. Table VI shows the comparison of detection results with different particles. Table VII describes the initial class centers and the final class center after using PSO optimization of the first particle.

In addition, the performance of the IDCPSO algorithm also depends on the value of K; we varied K's value from 2 to 10.Fig.12 shows the ROC curves for 5 different K values. For this particular data set, K=8 is a better choice than other values in that the attack detection rate nearly reaches 98% and the false positive rate remains as low as 2%. The larger the K's value, the longer the algorithm's computing time. Therefore, we

can find compromise between the detection rate and the computing time when K equal to 8.

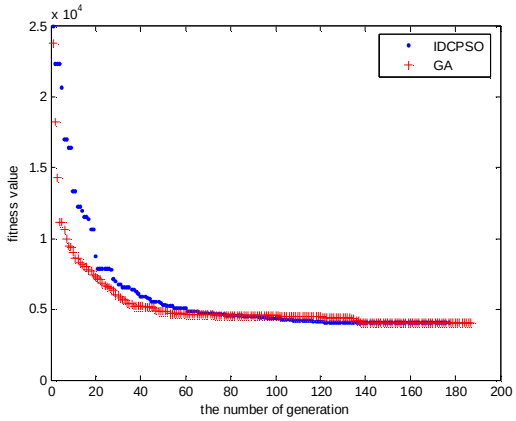


Figure 9. The comparison of fitness function results

TABLE V.
THE COMPARISON OF PERFORMANCE OF IDCPSO AND GA

	IDCPSO	GA
DR	0.967	0.933
FPR	0.023	0.171
FNR	0.033	0.066
Fitness function	4051.4	4239.2
The number of generations	176	190

TABLE VI.
THE COMPARISON OF DETECTION RESULTS WITH DIFFERENT PARTICLES

		DR	FPR	FNR
GA	particles =10	0.924	0.040	0.080
	particles =5	0.048	0.988	0.960
IDCPSO	particles =10	0.972	0.022	0.020
	particles =5	0.972	0.022	0.020

TABLE VII.
THE INITIAL CLASS CENTER AND FINAL CLASS CENTER OF THE FIRST PARTICLE

	The initial class center		The final class center	
	Normal	Attack	Normal	Attack
1	0.017607	-0.018325	-0.5400	1.5776
2	-0.098705	0.10273	0.1012	-0.2957
3	0.009729	-0.010126	-0.5771	1.6861
4	0.010499	-0.010928	-0.5769	1.6855
5	0.034638	-0.036052	-0.4478	1.3082
6	0.12934	-0.13462	-0.1104	0.3224
7	0.004035	-0.004200	-0.5290	1.5454

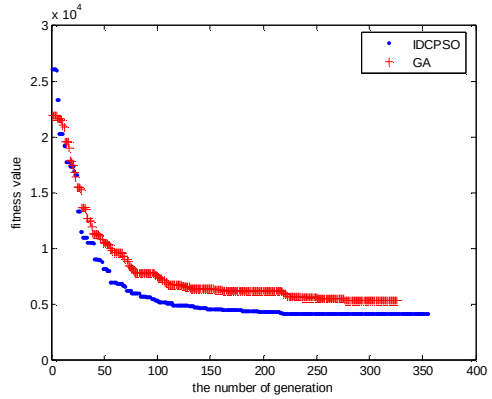


Figure 10. The convergence process of fitness function with particles=10

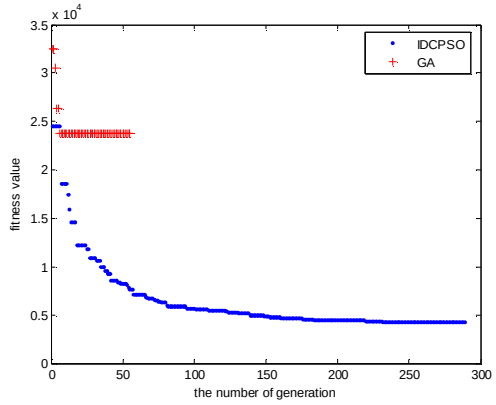


Figure 11. The convergence process with particles=5

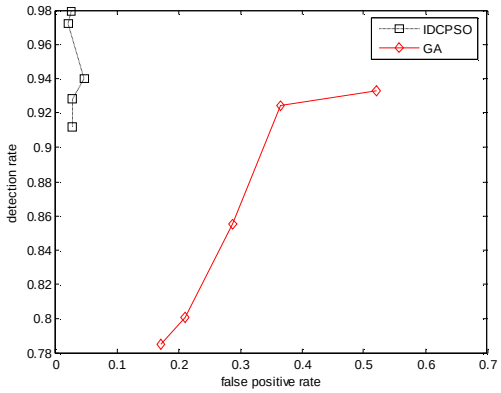


Figure 12. the ROC curve of IDCPSO and GA with different K

V. CONCLUSIONS

Prevention of security breaches completely using the existing security technologies is unrealistic. As a result, intrusion detection is an important component in network security. In this paper, IDCPSO has been proposed which combines the clustering with PSO algorithm. The simulation experiments compared with GA, indicate the following aspects:(1)we can obtain the global optimal value by using the IDCPSO algorithm;(2) detection rate is higher and FPR, FNR are lower than that of GA;(3)the speed of the convergence process is also higher than that of GA.

Our future work will focus on how to improve the detection rate on predicting attacks, especially the attacks of U2R and R2L. And due to the parameters' impact on the algorithm, we will consider introducing the other intelligent algorithms for the optimal parameters' selection.

ACKNOWLEDGEMENTS

The work in this paper is supported by the Natural Science Foundation of Chongqing, China (Grant No. 2008BB2182 and 2008BB0173), the Innovation Ability Training Foundation of Chongqing University, China (Grant No. CDCX021), and the National Natural Science Foundation of China (Grant No. 61070246).

REFERENCES

[1] Yuebin Bai, Hidetsune Kobayashi. Intrusion detection systems:technology and development.Proceedings of the17 th International Conference on Advanced Information Networking and Applications (AINA'03).

[2] L. Portnoy,E. Eskin,S. Stolfo. Intrusion detection with unlabeled data using clustering. In:Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001) Philadelphia. 2001:5-8.

[3] Nam Hun Park, Sang Hyun Oh, Won Suk Lee. Anomaly intrusion detection by clustering transactional audit streams in a host computer. Information Sciences 180 (2010) 2375–2389.

[4]Shi-Jinn Horng, Ming-Yang Su, Yuan-Hsin Chen, Tzong-Wann Kao, Rong-Jian Chen, Jui-Lin Lai,Citra Dwi Perkasa. A novel intrusion detection system based on hierarchical clustering and support vector machines. Expert Systems with Applications 38 (2011) 306–313.

[5] Haidong Yang, Chunsheng Li , Jue Hu. RFID intrusion detection with possibilistic fuzzy c-Means clustering.Journal of Computational Information Systems, v 6, n 8, p 2623-2632, August 2010.

[6] Liang Hu, Nurbol, Xiaobo Liu, Kuo Zhao. A time stamped clustering method for intrusion detection. Journal of Information and Computational Science, v 7, n 2, p 399-406, February 2010.

[7]Panda, Mrutyunjaya,Patra, Manas Ranjan. A hybrid clustering approach for network intrusion detection using cobweb and FFT. Journal of Intelligent Systems, v 18, n 3, p 229-245, 2009.

[8] Pavel Laskov, Patrick D¨ussel, Christin Sch¨afer and Konrad Rieck. Learning intrusion detection:supervised or unsupervised?. 12489 Berlin, Germany.

[9]Y. G. Liu, K. F. Chen, X. F. Liao, Wei Zhang. A genetic clustering method for intrusion detection. Pattern Recognition. 37 (2004):927–942.

[10]Y. H. Liao, V. R. Vemuri. Use of K-nearest neighbor classifier for intrusion detection. Computers Security 2002;21:439–448.

[11]C. F. Tsai, C. Y. Lin. A triangle area based nearest neighbors approach to intrusion detection. Pattern Recognition, In Press, Corrected Proof, Available online 3 June 2009.

[12] W. H. Chen, S. H. Hsu, H. P. Shen. Application of SVM and ANN for intrusion detection. Computers & Operations Research, Volume 32, Issue 10, October 2005: 2617-2634.

[13] WE1 Yu-xin, WU Mu-qing. KFDA and clustering based Detection. The journal of china universities of posts and telecommunications. Volume 15, Issue 1, March 2008, pages: 123-128.

[14] Yang Yi, Jiansheng Wu, Wei Xu. Incremental SVM based on reserved set for network intrusion detection. Expert Systems with Applications, 38 (2011):7698-7707.

[15] Yao, Yu, Yang, Wei; Gao, Fu-Xiang; Yu, Ge. Anomaly intrusion detection approach using hybrid MLP/CNN neural network. In proceedings - ISDA 2006: Sixth International Conference on Intelligent Systems Design and Applications, 2006, v2, pages:1095-1102.

[16] Xiaobai Li. A scalable decision tree system and its application in pattern recognition and intrusion detection. Decision Support Systems 41 (2005):112–130.

[17] M. Amini, R. Jalili, H. R. Shahriari. RT-UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks. Computers & Security, Volume 25, Issue 6, September 2006:459-468.

[18] Günes Kayacik, H. Nur Zincir-Heywood, A.; Heywood, Malcolm I. On the Capability of an SOM based Intrusion Detection System. Proceedings of the International Joint Conference on Neural Networks, v 3, pages:1808-1813, 2003.

[19] H.G. Kayacik, A.N. Zincir-Heywood, M.I. Heywood, A hierarchical SOM-based intrusion detection system, Engineering Applications of Artificial Intelligence 20 (4) (2007) 439–451.

[20] X. B. Tan, H. S. Xi. Hidden semi-Markov model for anomaly detection. Applied Mathematics and Computation, Volume 205, Issue 2, 15 November 2008:562-567.

[21] S. H. Oh, W. S. Lee. An anomaly intrusion detection method by clustering normal user behavior. Computers & Security. 2003.22(7): 596-612.

[22] K. Leung, et al. Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters. Proceedings of the Twenty-eighth Australasian conference on Computer Science, 2005:333-342.

[23] L. Khan, M. Awad, B. Thuraisingham, A new intrusion detection system using support vector machines and hierarchical clustering, The VLDB Journal 16 (2007): 507–521.

[24] X. Cheng, P. C. Yong, L. S. Meng. Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees. Pattern Recognition Letters. Volume 29 , Issue 7 (May 2008) :918-924.

[25] Hongying Zheng, Meiju Hou, Yu Wang. Application of Particle Swarm Optimization to Clustering for Intrusion Detection. The proceeding of 3rd international symposium on parallel architectures, algorithms and programming. Dalian, China,18-20 december 2010,Pages:221-228.

[26]D. E. Brown. C. L. Huntley. A Practical Application of Simulated Annealing to Clustering. Pattern Recognition,1992, 25(4):401-412.

[27]R. Eberhart, J. Kennedy. A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micromachine and human science. Nagoya. 1995:39-43.

[28]J. Kennedy, R. Eberhart. Particle swarm optimization. In: Proceedings IEEE international conference on neural networks. Perth. 1995.1942-1948.

[29]Y.S. Jiang, J.X. Wang, H. Z. Yang. Attribute Discretization for Decision System Based on Binary Particle Swarm Optimization. Control Engineering of China:2008 V0l.15, No.4:360-363.

[30]C. J. Liao, C. T. Tseng, P. Luarn. A discrete version of particle swarm optimization for flowshop scheduling problems. Computers & Operations Research, Volume 34, Issue 10, October 2007:3099-3111.

[31]M. Maitra, A. Chatterjee .A hybrid cooperative–comprehensive learning based PSO algorithm for image segmentation using multilevel thresholding. Expert Systems with Applications, Volume 34, Issue 2, February 2008:1341-1350.

[32]H. Pan, L. Wang, B. Liu. Particle swarm optimization for function optimization in noisy environment. Applied Mathematics and Computation, Volume 181, Issue 2, 15 October 2006: 908-919.

[33]T. K. Rasmussen, T. Krink. Improved Hidden Markov Model training for multiple sequence alignment by a particle swarm optimization—evolutionary algorithm hybrid. Biosystems, Volume 72, Issues 1-2, November 2003:5-17.

[34]P. Y. Yin, S. S. Yu, P. P. Wang, Y. T. Wang. A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems. Computer Standards & Interfaces 28 (2006): 441–450.

[35]Lincoln Labs, KDD-cup data set. http://kdd.ics.uci.edu/databases/kddcup99.html.

[36]Louis Gosselin, Maxime Tye-Gingras, François Mathieu-Potvin. Review of utilization of genetic algorithms in heat transfer problems. International Journal of Heat and Mass Transfer, Volume 52, Issues 9-10, April 2009, Pages 2169-2188.

[37] N.F. Wang, K. Tai. Target matching problems and an adaptive constraint strategy for multiobjective design optimization using genetic algorithms. Computers & Structures, Volume 88, Issues 19-20, October 2010, Pages 1064-1076.

[38]Xiao-Ping Zeng, Yong-Ming Li, Jian Qin. A dynamic chain-like agent genetic algorithm for global numerical optimization and feature selection. Neurocomputing, Volume 72, Issues 4-6, January 2009, Pages 1214-1228.

BIOGRAPHIES

Hongying Zheng is an associated professor in the College of Computer Science, Chongqing University in Chongqing, P.R. China. She was born in 1975. She received her B.S. degree in application of computer and M.S. degree in architecture of computer from chongqing university in 1999 and 2002 respectively, and Ph.D. degree was received in information security. Her research interests mainly include information security, neural network, data mining and etc.

Meiju Hou was born in Henan,China,in 1987.She received the B.S. degree in computer science and technology from Zhengzhou university, Henan, China, in 2009. At present, she is working on M.S. degree in computer science, Chongqing university, Chongqing,China. Her main research interests focus on the application of computational intelligence in intrusion detection systems.

Yu Wang was born in Chongqing,China,in 1987. He received his Batcher's Degree from the Chinese People's Liberation Army Artillery Institute, Hefei, China in 2004. He is currently a postgraduate in the College of Computer Science of Chong University, Chongqing, China. His research interests include database system architecture and information security.

A Hybrid Method for XML Clustering by Structure and Content

Yong Piao and Xiu-kun Wang
School of Electronic and Information Engineering, Dalian University of Technology, Dalian, China
Email: {piaoy, jsjwxk}@dlut.edu.cn

***Abstract*—An effective XML cluster method called neighbor center clustering algorithm (NCC) is presented in this paper, whose similarity is obtained through both structural and content information contained in XML files. Structural similarity is firstly measured by frequency-path model and its similarity calculation algorithm with position and frequency weight by longest common subsequence is introduced. In order to improve the performance and precision, the frequency-path model is further extended by considering the structure and content information simultaneously. Experiments show that the NCC embed with hybrid similarity calculation method can obtain high purity and F-measure value and is effective and applicable for clustering XML with both homogenous and heterogeneous structures.**

***Index Terms*—neighbor center clustering, position and frequency weight, longest common subsequence, hybrid similarity calculation**

I. INTRODUCTION

XML (eXtensible Markup Language), as a common data representation and exchange format on the Internet, contains a rich entailment of information. Also, data mining on XML has become an important part in text mining research, in which large-scale text clustering is one of the effective solutions for massive texts. An efficient and fast XML clustering mechanism, which can provide better data for decision support, will greatly shorten the information retrieval time, improve the efficiency of data query and help find out potential value of information. The most important feature of XML document, which is different from other textural ones, is its structural character. For this reason, we believe that the structure of XML should also play important role in XML clustering.

Considering the structural character of XML documents, many traditional text clustering methods are not suitable for XML. Currently, partitioning and hierarchical methods are most widely used in XML clustering [1-3], but the effect of these two traditional methods in dealing with irregular non-spherical document clustering is not so satisfactory, and besides, they are not well in distinguishing noise or isolated points effectively. In terms of computational complexity, the searching time of traditional methods for cluster centers increase rapidly, this is an obstacle to get better performance in XML clustering. In addition, traditional partitioning methods represented by K-Means and K-Medoids have to be specified the clusters number K in advance. Due to these reasons, a neighbor center clustering algorithm with similarity (NCC) is proposed in this situation. It is not only simple, but can find non-spherical structure documents and distinguish noise or isolated point effectively as well.

Similarity among documents is the key issue in the field of document clustering. So far, the methods proposed for this purpose can be roughly classified into three types, namely by the graph matching, by the edit distance or by the tree path model. Reference [4] describes an XML document with a directed graph and calculates similarity between XML documents by graph matching in order to cluster XML with similar structure. But the result is not satisfactory due to its low accuracy. Reference [5] improves [4], in which its method leads to some limitations to clustering results without considering the order relationships between edges in the discussion of equal direct edges. Reference [6] and [7] introduce a concept of edit distance. Reference [6] calculates the similarity with graph matching algorithm by describing XML as a directed graph, while [7] uses tree editing distance to calculate the similarity, so do [1,8,9]. However, it is not suitable for document processing due to its high computational complexity. As the graph cannot express XML structure well, Reference [10] proposes a tree path model representation, which is simpler than the tree editing distance with a lower time complexity, but it uses a complete path matching method widely while handling the matching procedures, so do references [11,12] in frequent path mining and matching, including its improvement [13]. The complete path matching method is useful in XML clustering in tree path models because XML structure information ignored by the complete path matching has little impact on clustering XML which have the same DTD, but it is not true if their DTDs are different, e.g. they have various structures.

In this paper our similarity measurement among XML documents with different structures is firstly presented,

Corresponding author: Yong PIAO

doi:10.4304/jsw.6.12.2361-2368

which is a similarity calculation algorithm with position and frequency weight by longest common subsequence (PFWLCS). On the basis of expressing XML document structures by correspondent paths using DOM tree, we extend the original tree path model to the frequency-path model by which we not only preserve label information of correspondent nodes, which decrease the original tree path model scale consumedly on the condition of not losing meaningful information and reduce the burden for later calculation, but also save the frequency structure of the original XML to improve the accuracy of the similarity. It makes the calculated similarity closer to the actual value by using the longest common subsequence method with position and frequency information. Furthermore, we continue to improve the frequency-path model by also considering node textural content, which makes the result more accurate and applicable, e.g. the hybrid method.

In Section II, Frequency-Path model and basic idea of hybrid similarity calculation methods are briefly introduced, followed by main steps of cluster algorithm NCC in detailed description in Section III. In Section IV experiments and its results are given showing better performance of our methods. Finally we summarize the whole work and provide future applications and research directions.

II. Similarity Calculation

A. Structural Similarity

1) Frequency-Path Model

Definition 1: $FPath=(f, v_1, v_2, \ldots, v_m, c_1, c_2, \ldots, c_m)$, where f denotes the number of occurrences of path in the current document; $(v_1,v_2,\ldots,v_m)$ is a node sequence from the root of XML DOM tree to one of its leaves; the c_i in $(c_1,c_2,\ldots,c_m)$ denotes the number of occurrences of v_i, whose ancestor nodes $v_1,v_2,\ldots,v_{i-1}$ are the same; m denotes the length of FPath (the node sequence).

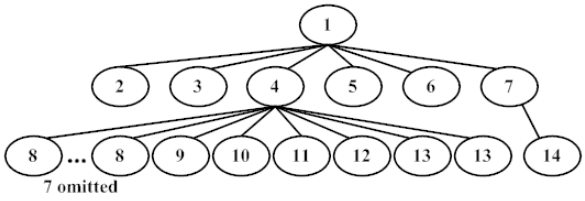
Definition 2: $XMLDoc=(FPath_1, FPath_2, \ldots, FPath_n)$, $FPath_i$ and $FPath_j$ are not the same FPath. We call two FPaths same only if the nodes at corresponding location of the two FPaths are completely identical. n denotes the number of various FPath.

Before calculating the similarity of XML, we extract the structure information of XML into FPaths where only the label of the node (structure) is considered. Other information such as data type and constraints are ignored. In Fig. 1 is an XML tree model and in Fig. 2 is a path model with its statistical information.

2) Frequency-Path Model Generation

Fig. 3 is the pseudo code of the algorithm to create the FPath model from XML document. The input of this algorithm is an XML document and output is an FPath model of it.

We also consider the semantic meanings that a node name can have during the structure matching. It is necessary since we are aiming at XML documents from different DTDs, which may not use the same word to express the similar meaning. For expressing the similar meaning only one word was taken from the synonymous word sets provided by WordNet. Besides, we assume that the node in higher hierarchy contributes more to the similarity than in lower hierarchy during the FPath matching. We will cover that in detail later.



1: automobile
2: manufacturer
3: model
4: feature
5: engine
6: year
7: engine
8: pro
9: mpg
10: safety
11: drive
12: seat
13: con
14: type

Figure 1. XML DOM Model

For reducing the complexity of getting semantic FPath, we use a parameter ζ to denote the depth of the node hierarchy being considered. For example, if $\zeta=1$, we just consider the meaning of the first node (root) of FPath.

```
1 automobile manufacturer 20 1
1 automobile model 20 1
1 automobile year 20 1
1 automobile engine type 20 1 1
1 automobile transmission 20 1
1 automobile feature safety 20 15 1
1 automobile feature drive 20 15 1
1 automobile feature seat 20 15 1
1 automobile feature mpg 20 15 1
9 automobile feature pro 20 15 9
2 automobile feature con 20 15 2
```

Figure 2. A frequency-path model

3) Algorithm PFWLCS

In this part the algorithm PFWLCS (Position and Frequency Weight by Longest Common Subsequence) used to calculate structural similarity of XML documents is described.

Definition 3: Subsequence: we call $<a_{i1}, a_{i2}, \ldots, a_{ik}>$ a subsequence of $<a_1, a_2, \ldots, a_n>$, only if $1<=i1<i2<\ldots<ik<=n$.

Definition 4: Common subsequence: we call $<c_1, c_2, \ldots, c_k>$ one common subsequence of $<a_1, a_2, \ldots, a_n>$ and $<b_1, b_2, \ldots, b_m>$, only when $<c_1, c_2, \ldots, c_k>$ is a subsequence of $<a_1, a_2, \ldots, a_n>$ and also a subsequence of $<b_1, b_2, \ldots, b_m>$, k denotes the length of the common subsequence $<c_1, c_2, \ldots, c_k>$.

In our method, we use Longest Common Subsequence (LCS) in matching two given paths. In Table I different situations are illustrated by xPath [10-12], PCXSS [13] and LCS[14-15] respectively, showing more information can be kept by using LCS than other methods.

```
Name: getMypathModel
Input:  an XML document to: xmlDocumentName
        node hierarchy: ζ
Output: a text document stored F_Path model: mypathModel
Pseudo code: void getMypathModel
                    (xmlDocumentName, ζ, mypathModel.txt)
{doc=getDocument(xmlDocumentName);
 //Parse XML document
 root=getRoot(doc);//get doc root
 getMypathModel(root, mypathModel);
 //get the paths and store in mypathModel.txt

 //the semantic process
 Initialize wordList //for the semantic process
 while(ζ>0){
      for (each Path in mypathModel.txt){
           node=getNodeFromPath(ζ);//get the ζ node
           if(node exists in Line j of wordList)
                Use the first word of Line j in wordlist to
                 replace node;
      else
           Insert node and its synonymous words into
           a new line in wordlist;
      }
      ζ--;
 }}
```

Figure 3. FPath model generation

We introduce a position-frequency weight vector *[W(1), W(2), ..., W(n)]*, where *(1, 2, ..., n)* is the index of nodes in a FPath, representing the hierarchy level. We assume that the node in higher hierarchy contributes more to the similarity than in lower hierarchy during FPath matching. Then the weight function *W(i)* must be digressive with *i* increases. In addition, *W(i)* also has the character below.

$$W(i)>0, \sum W(i)=1 \ (i=1, \ldots, n)$$

TABLE I.
PATH MATCHING USING DIFFERENT METHODS

No.	$Path_1$	$Path_2$	LCS	PCXSS	xPath
1	(a,b)	(a,b,y)	(a,b)	(a,b)	NULL
2	(a,b)	(y,a,b)	(a,b)	(a,b)	NULL
3	(x,a,b)	(y,a,b)	(a,b)	(a,b)	NULL
4	(a,b,x)	(a,b,y)	(a,b)	NULL	NULL
5	(a,b,x)	(a,k,b,y)	(a,b)	NULL	NULL
6	(a,h,b,x)	(a,k,b,y)	(a,b)	NULL	NULL
7	(a,b,x,h)	(a,b,y,k)	(a,b)	NULL	NULL

To further explain the necessity for position-frequency weight vector, the followings are discussed.

Suppose there are several paths when calculating the XML document similarity: P_1*(1, a, b, c, 1, 1, 1),* P_2*(1, a, b, x, 1, 1, 1),* P_3*(1, a, x, b, 1, 1, 1),* P_4*(2, a, b, x, 2, 2, 1)*.

For the similarity comparison of actual data, the nodes in higher hierarchy have greater effect in the XML document tree, namely the more front position the node is at, the more contributions to the similarity during FPath matching. Therefore, the similarity between P_1 and P_2 is significantly higher than it between P_1 and P_3. It shows the weight function *W(i)* is closely related to the position factor, namely position weight *T(i)*.

We also notice that the similarity degree between P_1 and P_4 is significantly greater than it between P_1 and P_2. That is because in the case of FPaths with the same node position, higher frequency of the same node indicates its role is more dominant than other nodes. Hence, the weight function *W(i)* is also closely related to node's frequency factor, namely frequency weight *F(i)*.

From the above, the position-frequency weight function *W(i)* is composed of *T(i)* and *F(i)*, where the position weight function $T(i) = 1/2^i$. As for the frequency weight function *F(i)*, we first define frequency equation of node V_i as $c_i log(c_i/f_s+1)$ (c_i is from the definition 1; f_s is $\sum f$ for all FPaths in the document). Then the normalized function of frequency weight can be expressed as (1).

$$F(i) = c_i \log(c_i / f_s + 1) \Big/ \sum_{i=1}^{n} c_i \log(c_i / f_s + 1) \quad (1)$$

and the position-frequency weight function *W(i)* is,

$$W(i) = (T(i) + F(i))/2 \quad (2)$$

We prove that (2) satisfies the features of *W(i)*, e.g. $W(i)>0$ and $\sum W(i)=1$.

$$\sum_{i=1}^{n} W(i) = \frac{1}{2} \sum_{i=1}^{n} (T(i)+F(i))$$

$$= \frac{1}{2} \left(\sum_{i=1}^{n} T(i) + \sum_{i=1}^{n} F(i) \right)$$

$$= \frac{1}{2} \left(\sum_{i=1}^{n} \frac{1}{2^i} + \sum_{i=1}^{n} \frac{Ci \log(Ci/fs+1)}{\sum_{i=1}^{n} Ci \log(Ci/fs+1)} \right)$$

$$= \frac{1}{2} \left(\frac{\frac{1}{2}(1-\frac{1}{2^n})}{1-\frac{1}{2}} + 1 \right) = 1 - \frac{1}{2^{n+1}}$$

$$\therefore n \to \infty, \sum_{i=1}^{n} W(i) = \lim_{n \to \infty} (1 - \frac{1}{2^{n+1}}) = 1$$

Definition 5: FPath similarity: Suppose two FPaths, $FPath_1 = (f_{p1}, x_1, x_2, \ldots, x_n, c_{x1}, c_{x2}, \ldots, c_{xn})$, $FPath_2 = (f_{p2}, y_1, y_2, \ldots, y_m, c_{y1}, c_{y2}, \ldots, c_{ym})$, the longest common subsequence (LCS) is $LCSPath=(z_1, z_2, \ldots, z_k)$, the hierarchy of the nodes in LCSPath in $FPath_1$ is $Hierarchy_1=(l_1, l_2, \ldots, l_k)$ orderly, the hierarchy of the nodes in LCSPath in $FPath_2$ is $Hierarchy_2= (h_1, h_2, \ldots, h_k)$ orderly. Then the similarity of $FPath_1$ and $FPath_2$ is described as (3) below.

$$\text{similarity} = \left(\sum_{i=1}^{k} W(l_i) \Big/ \sum_{j=1}^{n} W(j) + \sum_{i=1}^{k} W(h_i) \Big/ \sum_{j=1}^{m} W(j) \right) \Big/ 2 \quad (3)$$

In some practical situations, the occurrence number of the same path is also an important component of XML structure information, but (3) does not contain the path frequency information and just uses label information, thus leading to the situation that the calculated result does not represent the true sense of XML similarity. So we integrate the path frequency into (3) in order to make the result closer to actual value.

From definition 5, f_{p1} and f_{p2} are the occurrence numbers of $FPath_1$ and $FPath_2$, while f_{p1}/f_{s1} and f_{p2}/f_{s2}

are the path frequencies of $FPath_1$ and $FPath_2$ (f_{s1} is the sum of all path frequency in $FPath_1$, and f_{s2} is the sum of all path frequency in $FPath_2$). On the basis of the TF-IDF statistics theory widely used in text mining, we assume that, the larger the recurrent number of a path in XML is, the higher path frequency it is. That is to say, the more important the path is, the more structure information the path contains. Therefore, the similarity of path should be appropriate to be improved, in which the path frequency is large and the degree of increasing similarity should be within the scope of $[1-\sum_i W(l_i)]/\sum_j W(j),\ i=1..k,\ j=1..n$, so (3) is modified as follows.

$$\text{similarity}=\left(\frac{\sum_{i=1}^{k}W(l_i)}{\sum_{j=1}^{n}W(j)}\left(1+\left(1-\frac{\sum_{i=1}^{k}W(l_i)}{\sum_{j=1}^{n}W(j)}\right)\frac{f_{p1}}{f_{s1}}\right)+\frac{\sum_{i=1}^{k}W(h_i)}{\sum_{j=1}^{m}W(j)}\left(1+\left(1-\frac{\sum_{i=1}^{k}W(h_i)}{\sum_{j=1}^{m}W(j)}\right)\frac{f_{p2}}{f_{s2}}\right)\right)\Big/2 \quad (4)$$

Definition 6: XML document similarity: Suppose two XMLDocs, $XMLDoc_1= (FPath_1, FPath_2, ..., FPath_n)$, $XMLDoc_2= (FPath_1', FPath_2', ..., FPath_m')$, $m>=n$, Each FPath in $XMLDoc_1$ finds LCS with every FPath in $XMLDoc_2$, and calculates the similarity according to (4). We denote the biggest similarity as s_i, the similarity between $XMLDoc_1$ and $XMLDoc_2$ is formed with (5) below.

$$\text{Similarity}=\left(\sum_{i=1}^{n}S_i\Big/n+\sum_{i=1}^{n}S_i\Big/m\right)\Big/2 \quad (5)$$

```
Name: PFWLCS
Input: two F_Path model: mypathModel_1, mypathModel_2
Output: similarity of two Documents
Pseudo code: double getPFWLCSSimilarity
                        (mypathModel_1, mypathModel_2)
{ //n1, n2: number of paths in mypathModel_1, mypathModel_2
 Initialize n1, n2;
 //fs1, fs2: sum of path frequencies in the two models
 Initialize fs1, fs2;
 if(n1>n2)
      return getSimilarity(mypathModel_2, mypathModel_1);
      else{
           for(each path p1_i in mypathModel_1){
                for(each path p2_j in mypathModel_2){
                //a1, a2: position-frequency weights of p1_i,p2_j
                Initialize a1, a2;
                //get the longest common path of p1_i,p2_j
                lcs=getLCS(p1_i,p2_j);
                //w1,w2: position-frequency weights of the lcs
                Initialize w1,w2;
                im1=w1/a1;
                sim2=w2/a2;
                //f1,f2: path frequency of p1_i,p2_j
                Initialize f1,f2;
                sim=(sim*(1+(1-sim1)f1/fs1)
                          +sim2*(1+(1-sim2)f2/fs2))/2;
                }
                similarity += sim;
           }
 }
 //get the similarity of documents
 similarity=(similarity/n1+similarity/n2)/2;
 return similarity;
}
```

Figure 4. PFWLCS algorithm

Fig. 4 is the pseudo code of the algorithm to calculate the similarity of XML document based on the FPath model. The input of this algorithm is two FPath models, and the output is the similarity of the two XML documents, from which the two inputted FPath models come.

B. Hybrid XML similarity calculation

The PFWLCS algorithm based on frequency and path is introduced previously. Although it can obtain better result when processing XMLs from different DTDs, structured information is mainly concerned, e.g. structural similarity without considering node content. We know that the content feature of XML structure has also effects on XML document classification and clustering and has contributions to the result of data mining, which should not be ignored.

The content information is the textual part between tags. A method called SCSC (Similarity Calculation with Structure and Content) is presented in this subsection. The Frequency-path model is firstly improved in SCSC by adding the element content vector under the same path, making the presentation of XML documents richer. Moreover, a level ratio is introduced in XML similarity calculation, considering both the structural and element content similarity and making the result more sensible.

1) Improved frequency-path model

Definition 7: Improved tree path model: $IFPath=(f, v_1,v_2,...,v_m, /E/,c_1,c_2,...,c_m)$, where f denotes the number of occurrences of path in the current document; $(v_1,v_2,...,v_m)$ is a node sequence from the root of XML DOM tree to one of its leaves; E is a content vector $(e_1,e_2,...,e_t)$ under structural path $(v_1,v_2,...,v_m)$; the c_i in $(c_1,c_2,...,c_m)$ denotes the number of occurrences of v_i, whose ancestor nodes $v_1,v_2,...,v_{i-1}$ are the same; m denotes the length of IFPath (the node sequence).

```
1 automobile manufacturer / Cadillac / 20 1
1 automobile model / Cadillac CTS / 20 1
1 automobile year /2004 / 20 1
1 automobile engine type / 255 hp / 20 1 1
1 automobile transmission / Speed Manua / 20 1
1 automobile feature safety
                / Driver-Passenger airbags / 20 15 1
1 automobile feature drive / RWD / 20 15 1
1 automobile feature seat / 5 / 20 15 1
1 automobile feature mpg / City / 20 15 1
9 automobile feature pro
                / ABS Air Base CD Player Leather
                  Seats Side Theft Tracking
                  Traction Control Highway / 20 15 9
2 automobile feature con
                / Transmission Changer / 20 15 2
```

Figure 5. An improved FPath model

Example in Fig. 5 is improved based on the model introduced before. A content vector $(e_1,e_2,...,e_t)$ is supplemented, by which the XML content information is

further kept, making hybrid computation of XML similarity of both structure and content possible.

2) Content similarity calculation

Suppose we have two paths P_1 and P_2 belonging to documents D_1 and D_2 respectively, E_1 is content vector of P_1 and E_2 is content vector of P_2.

Before processing, we use TF-IDF theory to complete feature word selection and extraction against E_1 and E_2, resulting in feature word vector E_1' and E_2'. Then we use cosine similarity equation to get content similarity.

$$\text{Similarity(content)} = \frac{\omega}{\sqrt{n1+n2}} \qquad (6)$$

Where n_1 is the dimension of E_1' and n_2 is the dimension of E_2', ω is the number of common feature words E_1' and E_2' have.

3) Overall Similarity Calculation

Due to the reason that a new part is added to the frequency-path model, the original similarity calculation method is about to be modified accordingly. Similarity of a complete document is defined:

$$Sim = \alpha * Sim_{content} + (1-\alpha) * Sim_{structure} \qquad (7)$$

The final similarity is determined by both structure and content similarity, where α is called the level ratio parameter, presenting the structure layer of the element content. Structural similarity is still calculated using PFWLCS algorithm.

The higher level an element is at, the more closely it is to root node and has more contribution. That is, the level ratio factor α is greater and so has close relation with the position factor $T(i)$:

$$\alpha = (T(i)+T(j))/2.$$

$T(i)$ and $T(j)$ are two position weights of the last nodes in the compared paths separately.

III. XML Documents Clustering

A. Steps of Algorithm NCC

1) A point is chosen as the initial cluster center O_1 from a data set.

2) Set the center threshold parameter $(\xi_1, \xi_1 \geq 0)$. Then the similarity of O_1 between each remaining points from the data set is calculated and compared. If the similarity is greater than ξ_1, the point will be put into the cluster C_1, where O_1 is in.

3) Set the neighbor threshold parameter $(\xi_2, \xi_2 \geq \xi_1)$. At this time, the similarity of the points except O_1 in C_1 with the remaining points from the data set is calculated and compared again. If the similarity is greater than ξ_2, the point will be put into the cluster C_1.

4) Set the isolated threshold parameter $(\varphi, \varphi=1, 2 \ldots n)$. If the number of points in C_1 is less than φ, the cluster C_1 will be discarded.

5) A point is chosen as another cluster center O_2 from the remaining points in the data set. Repeat *2) 3) 4)* steps until there are no points in the data set left.

```
Name: Similarity calculation using SCSC
Input: Two improved frequency-path models:
                    mypathModel_1, mypathModel_2
Output: Similarity score between the two documents
Process: double getSimilarity
                    (mypathModel_1, mypathModel_2)
{ Initialize n1, n2;
 //n1, n2: number of different paths in the two models
 if(n1>n2)
      return getSimilarity(mypathModel_2, mypathModel_1);
 else{
      //Extract element content feature words using TF-IDF
      mypathModel_1'=do_TFIDF(mypathModel_1);
      mypathModel_2'=do_TFIDF(mypathModel_2);
      for( each path p1_i in mypathModel_1'){
           for(each path p2_j in mypathModel_2'){
                //Get structural similarity using PFWLCS
                SimS=getPFWLCSSimilarity(p1_i, p2_j);
                //Get number of common feature words
                w=getCommonElet(p1_i, p2_j);
                //t1, t2: number of feature words
                Initialize t1, t2;
                //Calculate content similarity
                SimC=w/sqrt(t1+t2);
                //Ti, Tj: position weights of last nodes
                //in p1_i, p2_j
                Initialize Ti, Tj;
                //calculate level ratio factor
                a=getPostion();
                //calculate similarity between two paths
                sim=a*SimC (1-a)*SimS
           }
           similarity +=sim;
      }}
 similarity=(similarity/n1+similarity/n2)/2;
 return similarity;}
```

Figure 6. SCSC algorithm

B. Analyses of NCC

The basic idea of NCC algorithm is trying to identify the actual cluster from the data set in each iterative step greedily. Below explains the main idea of NCC algorithm.

Considering the XML document set $D=\{d_1, d_2, d_3, d_4, d_5, d_6\}$ (d_i denotes the i^{th} XML document), where d_1, d_2, d_3, d_4, d_5 are actually in the same cluster, while d_6 is in another one. Set the parameters $(\xi_1=0, \xi_2=0.1, \varphi=1)$.

NCC arbitrarily selects d_1 as the initial cluster center and then finds the similarity between d_1 and d_2 greater than ξ_1, so d_2 is put into the cluster C_1, where d_1 is. So does d_3. At the moment, in order to avoid the situation that the chosen cluster center may be irrelevant resulting in imperfect clusters, it calculates and compares the similarity of the points except d_1 in C_1, in this case d_2 and d_3, with the remaining points. The purpose here is to spread the function of cluster center out over the neighborhood points in one cluster, which tries the number of matched points into one cluster, e.g. C_1, as many as possible. In Fig. 7, d_4 and d_5 are also put into the cluster because of d_2 and d_3.

Compared with some traditional algorithms, the NCC algorithm reduces the repeated calculation complexity on choosing cluster center in each iterative step, and improves overall efficiency. The clustering result from Fig.

7 is non-spherical, and besides, isolated points or isolated clusters are also identified, e.g. the point d_6 in the Figure.

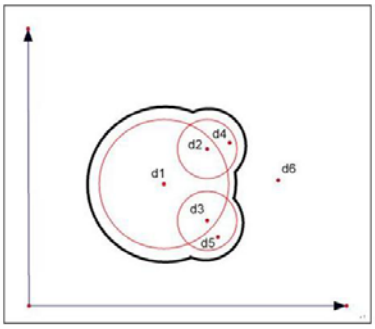


Figure 7. An Example of NCC clusters

NCC algorithm is suitable for XML clustering considering the XML structural character, while some vectorization methods ignore this kind of information during their calculation. It not only retains XML document's structural information, but also evaluates XML's textual meaning when with the method SCSC embedded. Therefore, the NCC algorithm based on SCSC has advantages in XML document clustering.

C. Evaluation of NCC

At present the two parameter indexes used widely to evaluate the overall performance of a clustering algorithm are the Purity and F-measure. The experimental data are the known document set or have usually been categorized before evaluation [16]. The purity of the cluster r is defined as follows:

$$P(Sr)=\frac{1}{n_r}*Max(n_r^i)$$

The purity of the overall clustering is defined as:

$$Purity=\sum_{r=1}^{k}\frac{n_r}{n}P(Sr)$$

Where n_r^i is the number of document belonging to type i, which is assigned to cluster r; n_r is the number of documents in cluster r; n is the number of the whole document set.

High purity can be easily achieved when the number of clusters becomes larger. In particular, the purity will be 1 if each document gets its own cluster. Thus, F-measure is introduced as a harmonic mean to combine both precision and recall factors.

Recall: $Recall(i,r) = n(i,r) / n_i$

Precision: $Precision(i,r) = n(i,r) / n_r$

where $n(i,r)$ is the number of document belonging to type i in the cluster r; n_r is the number of document in cluster r; n_i is the number of documents of type i. So the F-measure between cluster r and type i is defined as follows.

$$f(i,r)=\frac{2*recall(i,r)*precision(i,r)}{recall(i,r)+precision(i,r)}$$

The F-measure of the overall clustering is defined as:

$$F=\sum_{i}\frac{n_r}{n}Max\{f(i,r)\}$$

IV. Experiment and Discussion

The goal of our experiments is to examine the effectiveness of the SCSC by calculating the similarity of XML documents using both structure and content information. Further, SCSC is embedded into NCC in clustering XML and we obtained satisfactory results.

Two data sets are used in the following experiments. The first data set used in our experiment is a real life data set introduced in [17], which is generated in the XML/XSLT version of web pages from 20 different sites belonging to 4 categories, labeled as "automobile" , "movie" , "reference" and "software". There are a total of 120 documents: 24 in "automobile", 24 in "movie", 48 in "reference" and 24 in "software". There is no cross-labeling and the depth of the DOM tree of these XML documents is 5. The second data set is from Sigmod XML collection, 84 files and 4 DTDs are included.

In the experiments, NCC method is used to cluster the two data sets and evaluate results according the recall ratio, accuracy and F-measure introduced before.

First we use improved frequency-path model to extract information from all documents, then complete feature selection and extraction. We randomly choose one file from two data sets as cluster center and calculate similarities of other files against the cluster center respectively, this process is repeated 10 times and the result is as following:

Table II.
Experiment result

		DataSet1 (SCSC)	DataSet1 (PFWLCS)	DataSet2 (SCSC)
Average Values	Recall Ratio	81.13%	83.02%	88.56%
	Accuracy	94.08%	94.29%	95.81%
	F-measure	87.13%	88.30%	92.17%
	Purity	94.35%	93.89%	85.96%
	Clusters	3.72	3.80	3.87

From the Table II, results of dataset1 and dataset 2 are all satisfactory. Specifically, results of the first two columns, e.g. dataset1 (SCSC) considers both the structure and content information, while dataset1 (PFWLCS) is calculated by considering only structure information. Each item of dataset1 (SCSC) is a little lower than dataset1 (PFWLCS), that is because in dataset1 XML files are from many different DTDs, even in the same cluster. Therefore structural information plays more important role than content information in this situation. However, considering both the structure and content information is more nature and reasonable in practice and can reflect more real feature of data.

A good similarity calculation will make the similarity closer within one cluster and larger outside the cluster. In the following experiments, we pick up one file arbitrary from 4 clusters of the two datasets separately. We compare the similarity of this file to other files in its cluster and repeat this process 10 times. The following results are obtained.

TABLE III.
SIMILARITY COMPARISON IN DATASET1

$SimAvg(S_i,S_j)$	S1	S2	S3	S4
S1	0.4853	0.0054	0.0026	0.0018
S2	0.0054	0.4526	0.0021	0.0034
S3	0.0026	0.0021	0.4722	0.0012
S4	0.0018	0.0034	0.0012	0.4336

S_i is the i^{th} cluster in Dataset1, $SimAvg(S_i, S_j)$ is the average similarity between documents in cluster S_i and S_j.

TABLE IV.
SIMILARITY COMPARISON IN DATASET2

$SimAvg(Ci,Cj)$	C1	C2	C3	C4
C1	0.9646	0.1326	0.1464	0.1318
C2	0.1326	0.9524	0.1298	0.1431
C3	0.1464	0.1298	0.9346	0.1373
C4	0.1318	0.1431	0.1373	0.9546

C_i is i^{th} cluster in Dataset2, $SimAvg(C_i,C_j)$ is the average similarity of documents in clusters C_i and C_j.

From the two tables above, Similarities in the same cluster are all far greater than in other clusters using SCSC. We also notice that scores of dataset2 is a litter greater than in dataset1, this is because most XML files in dataset1 are from different DTDs and structure information become more significant, while in dataset2 XML files in one cluster are most from same DTD and content information become important. Anyway, SCSC can handle both of these conditions and results showing that the hybrid method is more applicable and effective.

V. CONCLUSION

We have demonstrated an XML document clustering method NCC, when embed with SCSC higher effectiveness will be achieved by considering both structure and content similarity. On one hand, the structural similarity is calculated using method PFWLCS with position frequency weight, based on frequency path model, in which more valuable information in XML clustering is kept using the longest common subsequence method and the position frequency weight vector. On the other hand, XML's content similarity is obtained through TF-IDF method. Experiments showed SCSC method could greatly help to improve clustering precision and performance, decrease complexity when embedded in NCC, and is suitable to both homogenous and heterogeneous XML documents.

REFERENCES

[1] T. Dalamagas, T. Cheng, K. J. Winkel, and T. Sellis, "A Methodology for Clustering XML Documents by Structure," *Information Systems*, vol. 31, pp. 187-228, 2006.

[2] G. Costa, G. Manco, R. Ortale, and A. Tagarelli, "A Tree-Based Approach to Clustering XML Documents by Structure," *Knowledge Discovery in Databases: PKDD 2004*, pp. 137-148, 2004.

[3] Pan Youneng, "Research on XML Document Cluster," *Journal of the China Society for Scientific and Technical Information*, 2006, 25(2).

[4] Wang Lian, David Wai-Lok Cheung, Nikos Mamoulis, and Siu-Ming Yiu, "An Efficient and Scalable Algorithm for Clustering XML Documents by Structure," *IEEE Transactions on Knowledge and Data Engineering*, 2004, 16(1), pp. 82-96.

[5] Liu Jiang and Wang Jun, "Reseach on Web XML Document Clustering," *Public Science*, 1002-6908(2007)0620038-03.

[6] Sudarshan S. Chawathe, Anand Rajaraman, Hector Garcia-Molina, and Jennifer Widom, "Change Detection in Hierarchically Structured Information," *In Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, 1996, pp. 493-504.

[7] Zhang K and Shasha D, "On the Editing Distance between Unordered Labeled Trees," *Information Processing Letters*, 1992, 42(3), pp. 133-139.

[8] A. Wojnar, I. Mlynkova and J. Dokulil, "Structural and Semantic Aspects of Similarity of Document Type Definitions and XML Schemas," *Information Sciences*, vol. 180, pp. 1817-1836, 2010.

[9] M. Torjmen, K. Pinel-Sauvagnat, and M. Boughanem, "Towards a Structure-based Multimedia Retrieval Model," *in 1st International ACM Conference on Multimedia Information Retrieval, MIR2008*, August 30, 2008 - August 31, 2008, Vancouver, BC, Canada, 2008, pp. 350-357.

[10] Sachindra Joshi, Neeraj Agrawal, Raghu Krishnapuram, and Sumit Negi, "A Bag of Paths Model for Measuring Structural Similarity in Web Documents," *SIGKDD'03*, 2003, pp. 24-27.

[11] YANG Hou-Qun, HE Zhong-Sh, and LEI Jing-Sheng, "Research of Clustering XML Documents Based on Partition," *Computer Science*, 2008, 35(3).

[12] Ho-pong Leung, Fu-lai Chung, Stephen C.F. Chan, and Robert Luk, "XML Document Clustering Using Common XPath," *In Proceedings of the 2005 International Workshop on Challenges in Web Information Retrieval and Integration, WIRI'05*, 2005, pp. 91-96.

[13] T. Tran and R. Nayak, "Evaluating the Performance of XML Document Clustering by Structure Only," *in 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006*, December 17, 2006 - December 20, 2006, Dagstuhl Castle, Germany, 2007, pp. 473-484.

[14] PIAO Yong, TIAN Wei, and WANG XiuKun, "An Effective Path-based Algorithm to Calculate XML Similarity," *Control and Decision*, 2010, 25(4), pp. 497-501.

[15] Y. Piao and X. K. Wang, "A Hybrid Method for XML Clustering," *in 3rd International Symposium on Parallel Architectures, Algorithms and Programming, PAAP 2010*, December 18-20, 2010, Dalian, China, 2010, pp.286-290.

[16] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, "An Introduction to Information Retrieval," Cambridge University Press, Cambridge, England, 2009, pp. 356-360.

[17] A. Kurt and T. Engin, "Classification of XSLT-generated Web Documents with Support Vector Machines," *In Knowledge Discovery from XML Documents*, 2006, pp.33-42.

[18] E. Bertino, G. Guerrini and M. Mesiti, "Measuring the Structural Similarity among XML Documents and DTDs,"

Journal of Intelligent Information Systems, vol. 30, pp. 55-92, 2008.
[19] E. Bertino, G. Guerrini and M. Mesiti, "A Matching Algorithm for Measuring the Structural Similarity between an XML Document and a DTD and its Applications," *Information Systems*, vol. 29, pp. 23-46, 2004.
[20] C. Wang, X. Yuan, H. Zhang, B. Sun, and H. Zhang, "Structural Query and Ranking for XML Information Retrieval," *Journal of Computational Information Systems*, vol. 5, pp. 1429-1435, 2009.
[21] H. Zhang, X. Yuan, N. Yang, and Z. Liu, "Similarity Computation for XML Documents by XML Element Sequence Patterns," Progress in WWW Research and Development, pp. 227-232, 2008.
[22] C. Wang, X. Yuan, H. Ning, and X. Lian, "Similarity Evaluation of XML Documents Based on Weighted Element Tree Model," *Advanced Data Mining and Applications*, pp. 680-687, 2009.
[23] A. Nierman and H. V. Jagadish, "Evaluating Structural Similarity in XML Documents," *in Proceedings of the Fifth International Workshop on the Web and Databases WebDB*, 2002.

Yong Piao, born in Liaoning province, China, 1975, received his B.A's and M.A's degrees in computer science from Dalian University and Technology, Dalian, Liaoning, China, in 1998 and 2001 respectively.

From 2002 to 2003, he made an advanced study in EI department, University of Siegen, Germany. Since 2004 he has been a lecturer, 2010 an associate professor, in Dalian University of Technology. His current research interests mainly cover database and decision support systems.

Xiu-kun Wang, born in 1945. She has been a professor in Dalian University of Technology. Her main research interests are data mining, database and decision support systems.

Hybrid Distributed Shared Memory Space in Multi-core Processors

Xiaowen Chen[†,‡], Shuming Chen[†], Zhonghai Lu[‡], Axel Jantsch[‡]
[†]Institute of Microelectronic and Microprocessor, School of Computer, National University of Defense Technology, 410073, Changsha, China
[‡]Department of Electronic Systems, School of Information and Communication Technology, KTH-Royal Institute of Technology, 16440 Kista, Stockholm, Sweden
[†]{xwchen,smchen}@nudt.edu.cn [‡]{xiaowenc,zhonghai,axel}@kth.se

Abstract— **On multi-core processors, memories are preferably distributed and supporting Distributed Shared Memory (DSM) is essential for the sake of reusing huge amount of legacy code and easy programming. However, the DSM organization imports the inherent overhead of translating virtual memory addresses into physical memory addresses, resulting in negative performance. We observe that, in parallel applications, different data have different properties (*private* or *shared*). For the private data accesses, it's unnecessary to perform Virtual-to-Physical address translations. Even for the same datum, its property may be changeable in different phases of the program execution. Therefore, this paper focuses on decreasing the overhead of Virtual-to-Physical address translation and hence improving the system performance by introducing hybrid DSM organization and supporting run-time partitioning according to the data property. The hybrid DSM organization aims at supporting fast and physical memory accesses for private data and maintaining a global and single virtual memory space for shared data. Based on the data property of parallel applications, the run-time partitioning supports changing the hybrid DSM organization during the program execution. It ensures fast physical memory addressing on private data and conventional virtual memory addressing on shared data, improving the performance of the entire system by reducing virtual-to-physical address translation overhead as much as possible. We formulate the run-time partitioning of hybrid DSM organization in order to analyze its performance. A real DSM based multi-core platform is also constructed. The experimental results of real applications show that the hybrid DSM organization with run-time partitioning demonstrates performance advantage over the conventional DSM counterpart. The percentage of performance improvement depends on problem size, way of data partitioning and computation/communication ratio of parallel applications, network size of the system, etc. In our experiments, the maximal improvement is 34.42%, the minimal improvement 3.68%.**

Index Terms— **Run-time Partitioning, Hybrid Distributed Shared Memory (DSM), Multi-core, processor**

I. INTRODUCTION

As a general trend, processor development has been shifted from single sequential processor to parallel multi-core systems [1] [2]. NoC based multi-core systems are promising solutions to the modern and future processor design challenges [3] [4] [5]. For instance, in 2007, Intel researchers announced their research prototype multi-core architecture containing 80 tiles arranged as a 10x8 2D mesh network [6]. In multi-core processors, especially for medium and large scale system sizes, memories are preferably distributed, featuring good scalability and fair contention and delay of memory accesses, since the centralized memory has already become the bottleneck of performance, power and cost [7]. In order to reuse huge amount of legacy code and facilitate programming, it's essential to support Distributed but Shared Memory (DSM). From the programmers' point of view, the shared memory programming paradigm provides a single shared address space and transparent communication, since there is no need to worry about when to communicate, where data exist and who receives or sends data, as required by explicit message passing API.

The key technique of Distributed Shared Memory organization is to maintain an address mapping table of translating virtual addresses into physical addresses and hence to implicitly access remote shared data and to provide software programmers with a transparent and global shared memory space. The Virtual-to-Physical address translation table (V2P Table) fully reveals how the DSM space is organized. However, the Virtual-to-Physical (V2P) address translation costs time, which is the inherent overhead of DSM organization. Every memory access operation contains a V2P address translation, increasing the system's processing time and hence limiting the performance. We observe that different data in parallel applications have different properties (*private* or *shared*) and it's unnecessary to introduce V2P address translations for private data accesses. According to the data property, we can obtain two observations:

(1) During the entire execution of parallel applications, some data processed by the local processor core are shared and need to be accessed by other remote processor cores, while other data are private and only

This paper is based on "Run-time Partitioning of Hybrid Distributed Shared Memory on Multi-core Network-on-Chips," by X. Chen, Z. Lu, A. Jantsch, and S. Chen, which appeared in the Proceedings of the 3rd International Symposium on Parallel Architectures, Algorithms, and Programming (PAAP), Dalian, China, December 2010. © 2010 IEEE.

The research is partially supported by the FP7 EU project MOSART (No. IST-215244), the Major Project of "Core electronic devices, High-end general purpose processor and Fundamental system software" in China (No. 2009ZX01034-001-001-006), the Innovative Team of High-performance Microprocessor Technology (No. IRT0614), the National Natural Science Foundation of China (No. 61070036), and the National 863 Program of China (No. 2009AA011704).

doi:10.4304/jsw.6.12.2369-2378

used by the local processor core.

(2) Some data are private and only accessed by the local processor core in a certain phase of the entire execution of parallel applications. However, they may change to be shared and accessible to other remote processor cores in another phase of the entire program execution.

The conventional DSM organizes all memories as shared ones, regardless of whether the processed data are only used by its local processor core or not. That is, each memory access operation in the system with conversional DSM organization includes a translation of virtual address to physical address, even if the accessed object is just local. If we get rid of the address translation overhead when the processor core handles the data which only belong to it (i.e. the data are *private*), the system's performance is expected to improve.

Motivated by aforementioned considerations, regarding the observation (1), we introduce a hybrid organization of Distributed Shared Memory in the paper. The philosophy of our hybrid DSM organization is to support fast and physical memory accesses for private data as well as to maintain a global and single virtual memory space for shared data. Considering the observation (2), we propose a run-time partitioning technique. This technique supports programmable boundary partitioning of private region and shared region of the Local Memory to change the hybrid DSM organization, based on the data property of parallel applications. It ensures fast physical memory addressing on private data and conventional virtual memory addressing on shared data, improving the performance of the entire system by reducing V2P address translation overhead as much as possible. We analyze its performance by formulating the run-time partitioning of hybrid DSM organization. The experimental results of real applications show that the hybrid DSM organization with run-time partitioning demonstrates performance advantage over the conventional DSM counterpart. The percentage of performance improvement depends on problem size, way of data partitioning and computation/communication ratio of parallel applications, network size of the system, etc. In our experiments, the maximal improvement is 34.42%, the minimal improvement 3.68%.

The rest of the paper is organized as follows. Section II discusses related work. Section III introduces our multi-core NoC platform and its hybrid DSM organization. Run-time partitioning is proposed in Section IV. Section V reports simulation results with application workloads. Finally we conclude in Section VI.

II. Related Work

As one form of memory organization, Distributed Shared Memory (DSM) has been attracting a large body of researches. However, we note that up to today there are few researches on NoC based multi-core chips. In [8], our previous work implemented a Dual Microcoded Controller to support flexible DSM management on multi-core processors. It off-loads DSM management from the main-processor to a programmable co-processor. In [9], Matteo explored a distributed shared memory architecture suitable for low-power on-chip multiprocessors. His work focused on the energy/delay exploration of on-chip physically distributed and logically shared memory address space for MPSoCs based on a parameterizable NoC. In our view, we envision that there is an urgent need to support DSM because of the huge amount of legacy code and easy programming. Monchiero also pointed out that the shared memory constitutes one key element in designing MPSoCs (Multiprocessor System-on-Chips), since its function is to provide data exchange and synchronization support [9]. Therefore, in the paper, we focus on the efficient organization and run-time partitioning of DSM space on multi-core processors.

Regarding memory partitioning, in [10], Xue explores a proactive resource partitioning scheme for parallel applications simultaneously exercising the same MPSoC system. His work combined memory partitioning and processor partitioning and revealed that both are very important to obtain best system performance. In [11], Srinivasan presented a genetic algorithm based search mechanism to determine a system's configuration on memory and bus that is energy-efficiency. Both Xue and Srinivasan addressed memory partitioning in combination with other factors, e.g. processors and buses. In [12], Mai proposed a function-based memory partitioning method. Based on pre-analysis of application programs, his method partitioning memories according to data access frequencies. Different from Xue's, Srinivasan's and Mai's work, this paper considers memory organization and partitioning according to data property of real applications running on multi-core processors. In [13], Macii presented an approach, called address clustering, for increasing the locality of a given memory access profile, and thus improving the efficiency of partitioning and the performance of system. In the paper, we improve the system performance by partitioning the DSM space into two parts: *private* and *shared*, for the sake of speeding up frequent physical accesses as well as maintaining a global virtual space. In [14], Suh presented a general partitioning scheme that can be applied to set-associative caches. His method collects the cache miss characteristics of processes/threads at run-time so that partition sizes are varied dynamically to reduce the total number of misses. We also adopt the run-time adjustment policy, but we address partitioning the DSM dynamically according to the data property in order to reduce Virtual-to-Physical (V2P) address translation overhead. In [15], Qiu also considered optimizing the V2P address translation in a DSM based multiprocessor system. However, the basic idea of his work is to move the address translation closer to memory so that the TLBs (Translation Lookaside Buffers: supporting translation from virtual to physical addresses) do not have consistency problems and can scale well with both the memory size and the number of processors. In the paper, we address reducing the total V2P address transaction overhead of the system as much

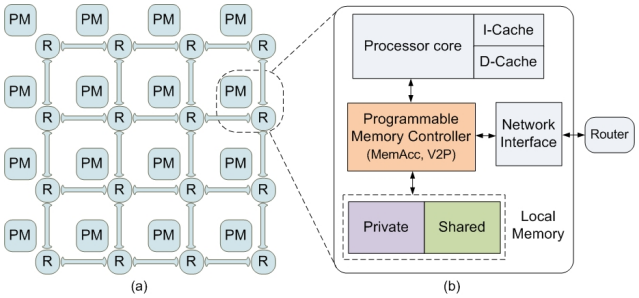


Figure 1. (a) Multi-core Processors, and (b) Processor-Memory Node

as possible by ensuring physical accesses on private data.

III. Multi-core Processors with Hybrid Distributed Shared Memory

In our multi-core processors, memories are distributed at network nodes but partly shared. Fig. 1 (a) shows an example of our multi-core processors with hybrid organization of Distributed Shared Memory. The system is composed of 16 Processor-Memory (PM) nodes interconnected via a packet-switched mesh network, which is a most popular NoC topology proposed today [16]. The microarchitecture of a PM node is illustrated in Fig. 1 (b). Each PM node consists of a processor core with tightly coupled caches, a network interface, a programmable memory controller and a local memory. As can be observed, memories are distributed in each node and tightly integrated with processors. All local memories can logically form a single global memory address space. However, we do not treat all memories as shared. As illustrated in Fig. 1 (b), the local memory is partitioned into two parts: *private* and *shared*. And two addressing schemes are introduced: *physical addressing* and *logic (virtual) addressing*. The private memory can only be accessed by the local processor core and it's physical. All of shared memories are visible to all PM nodes and organized as a DSM addressing space and they are virtual. For shared memory access, there requires a Virtual-to-Physical (V2P) address translation. Such translation incurs overhead. The philosophy of our hybrid DSM organization is to support fast and physical memory accesses for frequent private data as well as to maintain a global and single virtual memory space for shared data.

We illustrate the organization and address mapping of hybrid DSM in Fig. 2. As can be seen, a PM node may use both physical and logical addresses for memory access operations. Physical addresses are mapped to the local private memory region, and logical addresses can be mapped to both local shared and remote shared memory regions. For ***#k*** Node, its hybrid DSM space is composed of two parts. The first one is its private memory which is physical addressing. So the logic address is equal to the physical address in the private memory. The second part maps all shared memories. The mapping order of these shared memories is managed by the Virtual-to-Physical address translation table. Different PM nodes may have different hybrid DSM space. For instance, in the hybrid

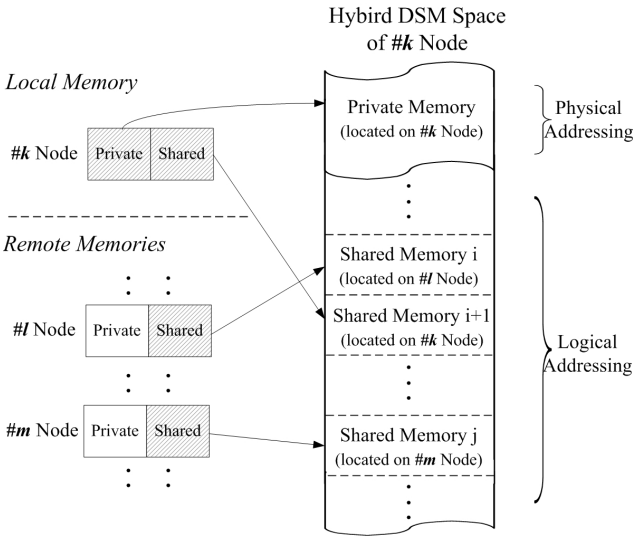


Figure 2. Hybrid organization of Distributed Shared Memory

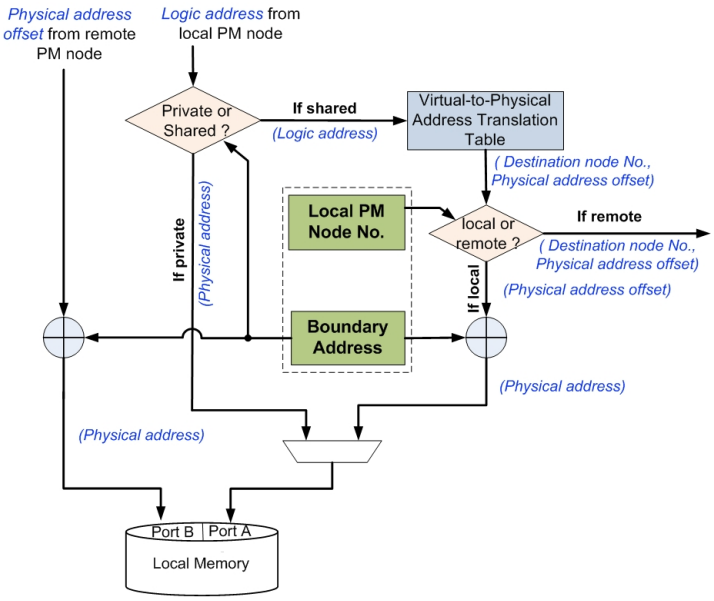


Figure 3. memory addressing flow

DSM space of ***#k*** Node, its local shared memory region is mapped logically as Shared Memory $i+1$ following Shared Memory i which is corresponding to the remote shared memory region in ***#l*** Node.

To support the hybrid DSM organization, the multi-core architecture maintains two registers for each PM node: **Local PM Node No.** and **Boundary Address** (see in Fig. 3). In each PM node, **Local PM Node No.** denotes the number of the local PM node, while **Boundary Address** denotes the address of boundary of the private region and the shared region in the Local Memory. **Boundary Address** can be configured at run-time to support dynamic re-organization of hybrid DSM space.

Fig. 3 shows the memory addressing flow of each PM node. As shown in the figure, each PM node can respond to two memory access requests concurrently from the local PM node and the remote PM node via the network.

In the beginning, the local PM node starts a memory access in its hybrid DSM space. The memory address is logic. The local PM node firstly distinguishes whether the address is private or shared. If private, the requested memory access hits the private memory. In the private memory, the logic address is just the physical address, so the address is forwarded directly to the Port A of the Local Memory. If the memory access is "write", the datum is stored into the Local Memory in the next clock cycle. If the memory access is "read", the data is loaded out of the Local Memory in the next clock cycle. The physical addressing is very fast. In contrast, if the memory address is shared, the requested memory access hits the shared part of the hybrid DSM space. The memory address first goes into the Virtual-to-Physical address translation table (V2P Table). The V2P Table records how the hybrid DSM space is organized and is responsible for translating the logic address into two pieces of useful information: *Destination Node No.* and *Physical Address Offset*. As shown in Fig. 2, the shared part of hybrid DSM space is composed by all shared memory regions of all PM nodes. *Destination Node No.* is used to obtain which PM node's shared memory is hit by the requested memory address. Once the PM node with the target shared memory is found, *Physical Address Offset* helps position the target memory location. *Physical Address Offset* plus **Boundary Address** in the target PM node equals the physical address in the target PM node. As shown in the figure, *Destination Node No.* and *Physical Address Offset* are obtained out of the V2P Table. Firstly, we distinguish wether the requested memory address is local or remote by comparing *Destination Node No.* with **Local Node No.**. If local, *Physical Address Offset* is added by **Boundary Address** in the local PM node to get the physical address. The physical address is forwarded to the Port A of the Local Memory to accomplish the requested memory access. If the requested memory access is remote, *Physical Address Offset* is routed to the destination PM node via the on-chip network. Once the destination PM node receives remote memory access request, it adds the *Physical Address Offset* by the **Boundary Address** in it to figure out the physical address. The physical address is forwarded to the Port B of the Local Memory. If the requested memory access is "write", the data is stored into the Local Memory. If the requested memory access is "read", the data is loaded from the Local Memory and sent back to the source PM node.

IV. Run-time Partitioning

Our multi-core processors support configuring **Boundary Address** at run-time in order to dynamically adjust the DSM organization when the program is running. Programmers can insert the following inline function in their C/C++ program to change the **Boundary Address** register.

```
void Update_BADDR(unsigned int Value);
```

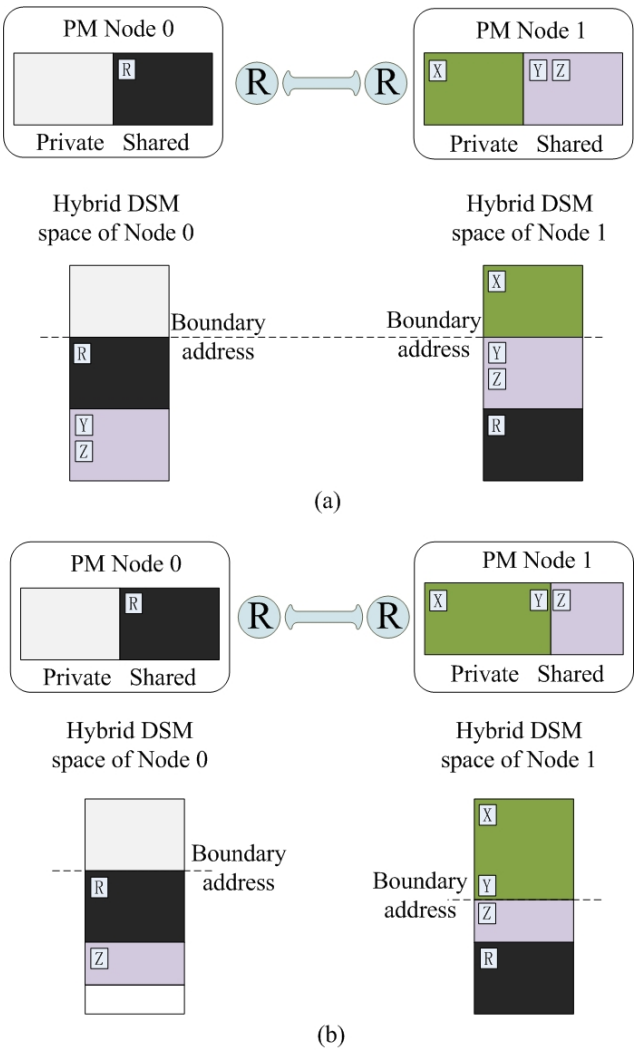


Figure 4. A short example of run-time DSM partitioning

A. Idea Description

Fig. 4 shows a short example of run-time partitioning of hybrid DSM space. Assume that there are two PM nodes in a 1x2 network. The hybrid DSM space of PM Node 0 contains its private memory region, its shared memory region and the shared memory region of PM Node 1, while the hybrid DSM space of PM Node 1 equals its Local Memory plus the shared memory region of PM Node 0. In the beginning (see Fig. 4 (a)), datum 'X' is in the private region of the Local Memory in PM Node 1, while datum 'Y' and 'Z' are in the shared region of the Local Memory in PM Node 1. Therefore, 'X' is invisible to PM Node 0, while 'Y' and 'Z' are accessible to PM Node 0. When PM Node 0 access 'Y', the *Destination Node No.* (i.e. PM Node 1) and *Physical Address Offset* of 'Y' are obtained from the V2P Table of PM Node 0 in the first phase. A remote request of memory access is sent to PM Node 1. Once PM Node 1 receives the request, the *Physical Address Offset* plus the **Boundary Address** in PM Node 1 equals the real physical address of 'Y'. Assume that the **Boundary Address** in PM Node 1 is re-configured to a larger value during the program

execution (i.e, the private memory part is enlarging) so that 'Y' becomes private. In this situation (see Fig. 4 (b)), PM Node 0 cannot access 'Y'. The procedure illustrated by Fig. 4 can be used to improve the system performance. For instance, PM Node 0 acts as a producer, PM Node 1 as a consumer. PM Node 0 produces several data which will be consumed by PM Node 1. In the beginning, the property of these data are *shared*, PM Node 0 firstly stores them into the shared region of the Local Memory of PM Node 1. After that, these data are only used by PM Node 1, it's unnecessary for PM Node 1 to access them in logic addressing mode. By changing the boundary address, we can make them be private. The Virtual-to-Physical address translation overhead is averted so that the system performance is improved.

The boundary address configuration is flexible to software programmers. When changing the boundary address, programmers need to pay attention to guarantee memory consistency. In the example of Fig. 4, changing the boundary address must be after PM Node 0 accomplish storing 'Y'. This can be guaranteed by inducing a synchronization point before PM Node 1 starts re-writing the **Boundary Address** register.

B. Producer-Consumer Mode

With concurrent memory addressing, the run-time partitioning provides software programmers with a *Producer-Consumer Mode*. The *Producer-Consumer Mode* features low-latency and tightly-coupled data transmission between two PM nodes. Concurrent memory addressing allows that one PM node produces data and in the meanwhile another PM node consumes the data that have been produced. During data transmission, the property of transmitted data may be changed. The run-time partitioning dynamically adjusts the DSM organization according to the changed data property and hence reduce V2P address translation overhead so as to improve the efficiency and performance of data transmission.

Fig. 5 illustrate the *Producer-Consumer Mode* under two cases. Assume that there are two PM nodes in a 2x1 network and m data blocks transferred from PM Node 0 to PM Node 1. The hybrid DSM space of PM Node 0 contains its private memory region, its shared memory region and the shared memory region of PM Node 1, while the hybrid DSM space of PM Node 1 equals its Local Memory plus the shared memory region of PM Node 0. The PM Node 0 produces m data blocks one by one, while the PM Node 1 consumes the m data blocks one by one once one of them are ready.

In Fig. 5, the m data blocks are located in the Local Memory of PM Node 1. As shown in Fig. 5 (a), all data blocks are with the property of "*shared*" since all data blocks haven't been produced by PM Node 0 yet. The **Boundary Address** indicates that B_1, ..., B_m are in the shared memory region of PM Node 1 and accessible to PM Node 0. In the beginning, PM Node 0 produces B_1. The values of B_1 are written into the shared memory region of PM Node 1. After PM Node 0 finishes producing B_1, there is a synchronization point that informs PM Node 1 of the readiness of B_1. Once B_1 are ready

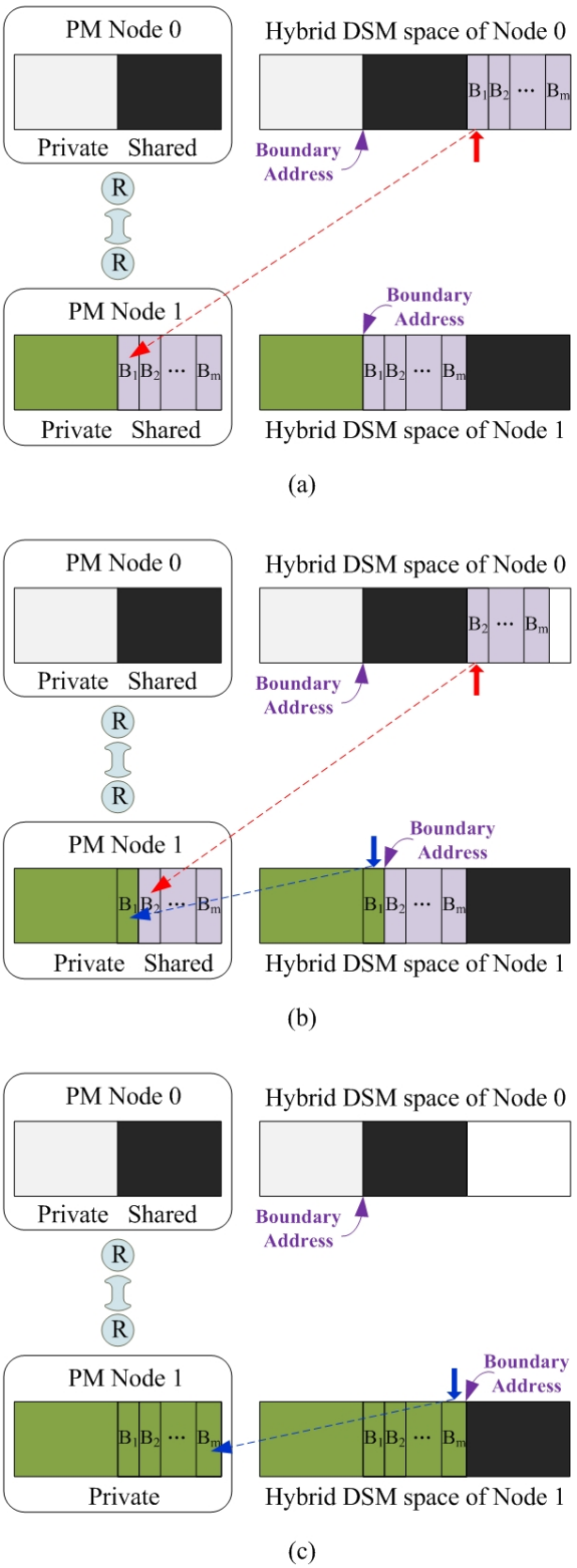


Figure 5. Producer-Consumer Mode

(see Fig. 5 (b)), PM Node 1 re-configures its **Boundary Address** to make B_1 be "*private*". After the run-time boundary address configuration, the hybrid DSM spaces of PM Node 0 and PM Node 1 are both changed. PM Node 0 only can access B_2, ..., B_m rather than B_1, and PM Node 1 can access B_1 in fast *physical addressing* scheme and hence eliminate V2P address translation overhead that would be induced in conventional DSM organization. The Programmable Memory Controller (see Fig. 1) allows concurrent memory references from both the local PM node and the remote PM node via the on-chip network. As shown in Fig. 5 (b), while PM Node 0 produces B_2, PM Node 1 is consuming B_1. The rest can be done in the same manner. So while PM Node 0 produces B_i, PM Node 1 is consuming B_{i-1} (i = 2,...,m). At last, PM Node 1 is consuming B_m, as shown in Fig. 5 (c). A this time, all data blocks becomes "*private*" and the entire Local Memory of PM Node 1 is private and invisible to PM Node 0. The hybrid DSM space of PM Node 0 only contains its Local Memory.

In Fig. 5, assume that the total transferred data size is fixed. Larger block size and hence smaller block number results in less overhead of run-time boundary address configuration but less concurrency. Less overhead of run-time boundary address configuration leads to positive performance, less concurrency negative performance. Therefore, there is a tradeoff to determine a optimized block size and boundary address configuration number.

C. Performance Analysis

Following the example shown in Fig. 4, we formulate the run-time partitioning of hybrid DSM organization of PM Node 1 and discuss its performance. To facilitate the analysis and discussion, we first define a set of symbols in TABLE I.

TABLE I.
DEFINITIONS AND NOTATIONS

N	data size processed by PM Node 1.
x	ratio of *private* data (e.g. 'X' in Fig. 4) to total data.
y	ratio of data with changeable property (e.g. 'Y' in Fig. 4) to total data.
z	ratio of *local shared* data (e.g. 'Z' in Fig. 4) to total data.
r	ratio of *remote shared* data (e.g. 'R' in Fig. 4) to total data.
t_{mem}	cycles of accessing the Local Memory once.
t_{v2p}	cycles of Virtual-to-Physical address translation.
t_{com}	cycles of network communication.
t_p	cycles of accessing a *private* datum. ($t_p = t_{mem}$)
t_{ls}	cycles of accessing a *local shared* datum. ($t_{ls} = t_{v2p} + t_{mem}$)
t_{rs}	cycles of accessing a *remote shared* datum. ($t_{rs} = t_{v2p} + t_{mem} + t_{com}$)
t_{bp}	cycles of changing the **Boundary Address** once.
m	number of boundary address configuration during program execution.

We firstly formulate the data access delay of conventional DSM organization. Memories are thought to be shared in conventional DSM organization and hence V2P address translation overhead is involved in every local or remote memory access. In Fig. 4, PM Node 1's accessing 'X', 'Y', 'Z', and 'R' includes V2P address translation overhead. Therefore, we can obtain its data access delay by Formula (1) below.

$$\begin{aligned} T_1 &= N \cdot (x + y + z) \cdot t_{ls} + N \cdot r \cdot t_{rs} \\ &= N \cdot t_{mem} + N \cdot t_{v2p} + N \cdot r \cdot t_{com} \end{aligned} \tag{1}$$

Formula (1) is subject to

$$x + y + z + r = 1$$

For our introduced hybrid DSM organization and its run-time partitioning, Virtual-to-Physical address translation overhead is only included when shared data are accessed. Therefore, in Fig. 4, PM Node 1's accessing 'X' and 'Y' doesn't need V2P address translation. However, the data access delay contains the extra overhead of changing the boundary address. We can get the data access delay by Formula (2) below.

$$\begin{aligned} T_2 &= N \cdot (x + y) \cdot t_p + N \cdot z \cdot t_{ls} + N \cdot r \cdot t_{rs} + t_{bp} \cdot m \\ &= N \cdot t_{mem} + N \cdot (z + r) \cdot t_{v2p} + N \cdot r \cdot t_{com} + t_{bp} \cdot m \end{aligned} \tag{2}$$

Formula (2) is subject to

$$\begin{cases} x + y + z + r = 1 \\ m \leqslant N \cdot y \end{cases}$$

Then, we can obtain the performance gain (γ: defined as average reduced execution time of accessing a datum) by Formula (3) below.

$$\gamma = \frac{T_1 - T_2}{N} = (x + y) \cdot t_{v2p} - \frac{t_{bp} \cdot m}{N} \tag{3}$$

From Formula (3), we can see that

(i) Compared with conventional DSM organization, our hybrid DSM organization gets rid of Virtual-to-Physical (V2P) address translation of private memory accesses and hence obtain performance improvement. Run-time partitioning further eliminate the V2P address translation of memory accesses of data whose data property is changeable, but it induces extra overhead of changing the boundary address.

(ii) If private data or data whose data property is changeable take a lager proportion in parallel programs (i.e. x and y increase), the hybrid DSM organization demonstrates higher performance advantage.

(iii) For the same size of parallel programs (x, y, z, and r are fixed) on hybrid DSM organization, hardware implementation of V2P address translation obtains higher performance gain than software implementation (t_{v2p} of hardware solution is greater than that of software solution).

(iv) The re-configuration of **Boundary Address** induces negative performance. However, avoiding frequently changing the boundary address results in little value of $\frac{t_{bp} \cdot m}{N}$ and hence alleviates its negative effect on performance.

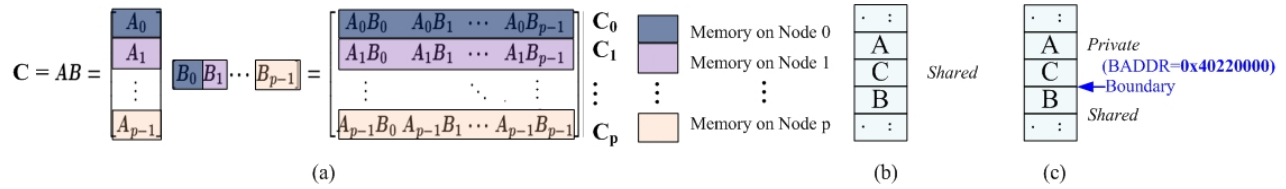


Figure 6. (a) Memory allocation for matrix multiplication, (b) conventional DSM organization, and (c) Hybrid DSM organization with run-time partitioning

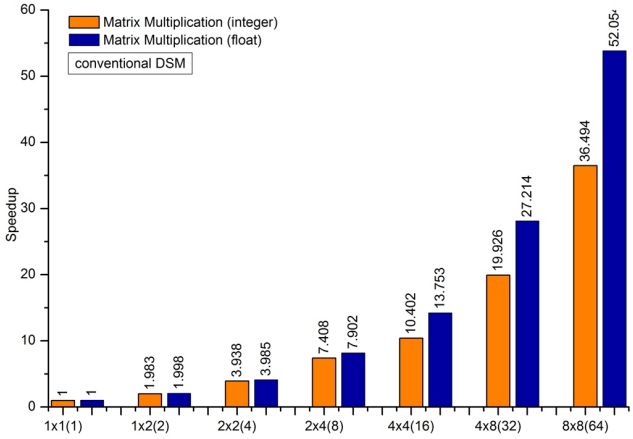


Figure 7. Speedup of matrix multiplication

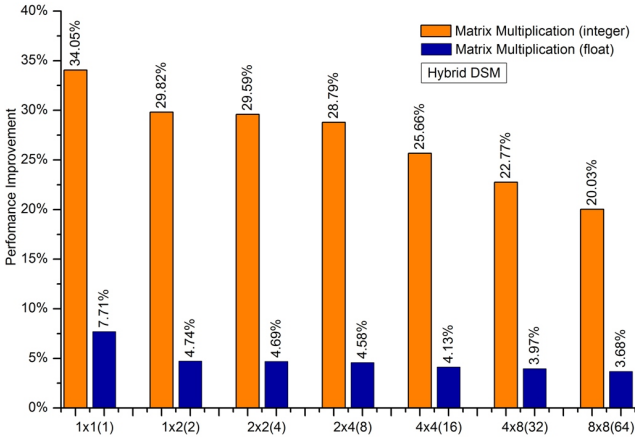


Figure 8. Performance improvement for matrix multiplication

V. Experiments and Results

A. Experimental Platform

We constructed a multi-core experimental platform as shown in Fig. 1. The multi-core processor uses the LEON3 [17] as the processor in each PM node and uses the Nostrum NoC [18] as the on-chip network. The LEON3 processor core is a synthesizable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture. The Nostrum NoC is a 2D mesh packet-switched network with configurable size. It serves as a customizable platform.

B. Application 1: Matrix Multiplication

The matrix multiplication calculates the product of two matrix, A[64, 1] and B[1, 64], resulting in a C[64, 64] matrix. We consider both integer matrix and floating point matrix. As shown in Fig. 6 (a), matrix A is decomposed into p equal row sub-matrices which are stored in p nodes respectively, while matrix B is decomposed into p equal column sub-matrices which are stored in p nodes respectively. The result matrix C is composed of p equal row sub-matrices which are respectively stored in p nodes after multiplication.

Fig. 6 (b) shows the conventional DSM organization and all data of matrix A, C and B are shared. Fig. 6 (c) shows the hybrid DSM organization. The data of matrix A and C are private and the data of matrix B are shared. Because the sub-matrices of matrix A and C are only accessed by their own host PM node and matrix B are accessed by all PM nodes. In this experiment, the system size increases by a factor of 2 from 1, 2, 4, 8, 16, 32 to 64. While mapping the matrix multiplication onto multiple cores, we perform a manual function partitioning and map the functions equally over the cores.

Fig. 7 shows the performance speedup of the matrix multiplication with conventional DSM organization. When the system size is increased from 1 to 2, 4, 8, 16, 32 and 64, the application speedup (speedup $\Omega_m = T_{1core}/T_{mcore}$, where T_{1core} is the single core execution time as the baseline, T_{mcore} the execution time of m core(s).) is from 1 to 1.983, 3.938, 7,408, 10.402, 19.926 and 36.494 for the integer computation, and from 1, 1.998, 3.985, 7.902, 13.753, 27.214, 52.054 for the floating point computation. The relative speedup for the floating point multiplication is higher than that for the integer computation. This is as expected because when increasing the computation time, the portion of communication delay becomes less significant, thus achieving higher speedup. Fig. 8 shows performance improvement of the hybrid DSM organization with respect to the conventional DSM organization. We calculate the performance improvement using the following formula:

$$\text{Perf. Impr.} = \frac{\text{Speedup}_{\text{hybrid DSM}} - \text{Speedup}_{\text{conventional DSM}}}{\text{Speedup}_{\text{hybrid DSM}}}$$

For the integer matrix, the performance increases by 34.05%, 29.82%, 29.59%, 28.79%, 25.66%, 22.77%, and 20.03%, for 1x1, 1x2, 2x2, 2x4, 4x4, 4x8 and 8x8 system size, respectively. For the floating point matrix, the performance is increased by 7.71%, 4.74%, 4.69%, 4.58%, 4.13%, 3.97%, and 3.68% for 1x1, 1x2, 2x2,

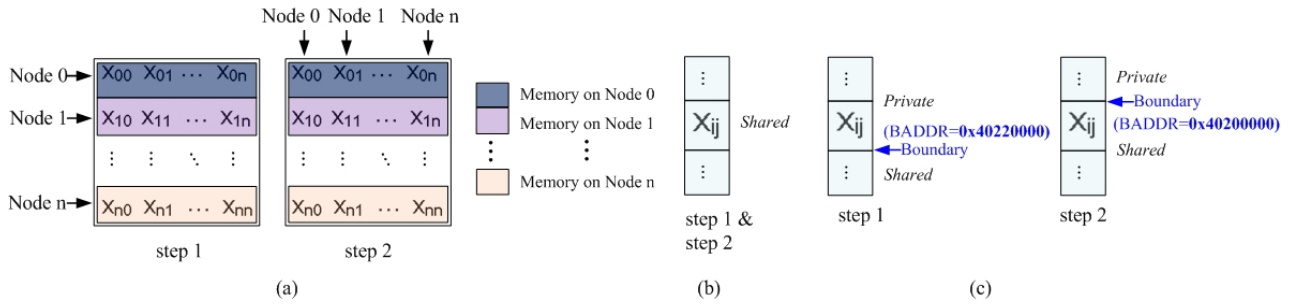


Figure 9. (a) Memory allocation for 2D DIT FFT, (b) conventional DSM organization, and (c) Hybrid DSM organization with run-time partitioning

2x4, 4x4, 4x8 and 8x8 system size, respectively. The improvement for the floating point is lower because the floating point has a larger percentage of time spent on computation, thus reducing the communication time in memory accesses achieves less enhancement. Note that, the single core case has a higher improvement because all data accesses are local shared for the conventional DSM organization and private for the hybrid DSM organization.

C. Application 2: 2D DIT FFT

We implement a 2D radix-2 DIT FFT. As shown in Fig. 9 (a), the FFT data are equally partitioned into n rows, which are stored on the n nodes, respectively. According to the 2D FFT algorithm, the application first performs FFT on rows (Step 1). After all nodes finish row FFT (synchronization point), it starts FFT on columns (Step 2). We experiment on two DSM organizations. One is the conventional DSM organization, as shown in Fig. 9 (b), for which all FFT data are shared. The other is the hybrid DSM organization, as illustrated in Fig. 9 (c). Since the data used for row FFT calculations at step 1 are stored locally in each node and are only to be used for column FFT calculations at step 2, we can dynamically re-configure the boundary address (BADDR in Fig. 9) at run time, such that, the data are private at step 1 but become shared at step 2.

Fig. 10 shows the speedup of the FFT application with the conventional DSM organization, and performance enhancement of the hybrid DSM organization with run-time partitioning. As we can see, when the system size increases from 1 to 2, 4,, 8, 16, 32, and 64, the speedup with the conventional DSM organization goes up from 1 to 1.905, 3.681, 7.124, 13.726, 26.153 and 48.776. The speedup for the hybrid DSM organization is normalized with the single core with the conventional DSM organization. As Fig. 10 shows, for the different system sizes, 1, 2, 4, 8, 16, 32 and 64, the performance improvement is 34.42%, 16.04%, 15.48%, 14.92%, 14.70%, 13.63% and 11.44%, respectively. Note that, the single core case has a higher improvement because all data accesses are local shared the conventional DSM organization and private for the hybrid DSM organization, and there is no synchronization overhead.

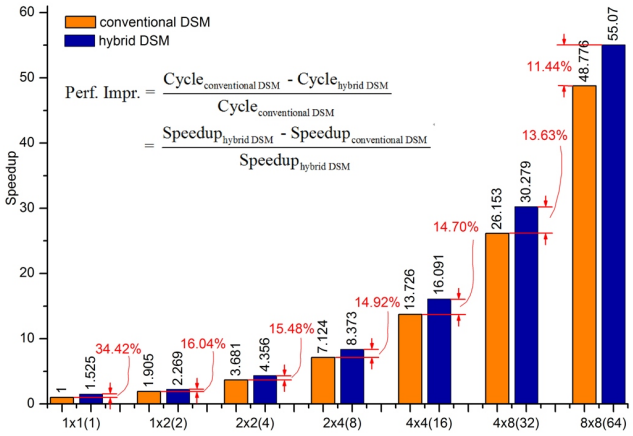


Figure 10. Speedup and performance improvement of 2D DIT FFT

D. Application 3: Wavefront Computation

Wavefront Computation is performed to validate the producer-consumer mode. Wavefront Computations are common in scientific applications. Given a matrix (see Fig. 11 (a)), the left and top edges of which are all 1, the computation of each remaining element depends on its neighbors to the left, above, and above-left. If the solution is computed in parallel, the computation at any instant form a wavefront propagating toward in the solution space. Therefore, this form of computation get its name as wavefront. We use the same method as [19] to parallelize the Wavefront Computation, the rows of the matrix are assigned to PM nodes in a round-robin fashion (see Fig.11 (b) and (c)). With this static scheduling policy, to compute an element, only the availability of its above neighbor needs to be checked (synchronized). For instance, PM node 0 computes the elements in row 1. PM node 1 cannot compute the elements in row 2 until the corresponding elements in row 1 has been figured out by PM node 0. After finishing the computation in row 1, PM node 0 goes on to compute the elements in row 3 according to the round-robin scheduling policy. In our experiment, we conduct a 64×64 matrix in two DSM organization.

For conventional DSM organization, as shown in Fig. 11 (b), all elements are *shared* and memory accesses on them incur V2P address translation overhead. For hybrid DSM organization, we apply *Producer-Consumer Mode* described in Subsection IV-B. As Fig. 11 (c)

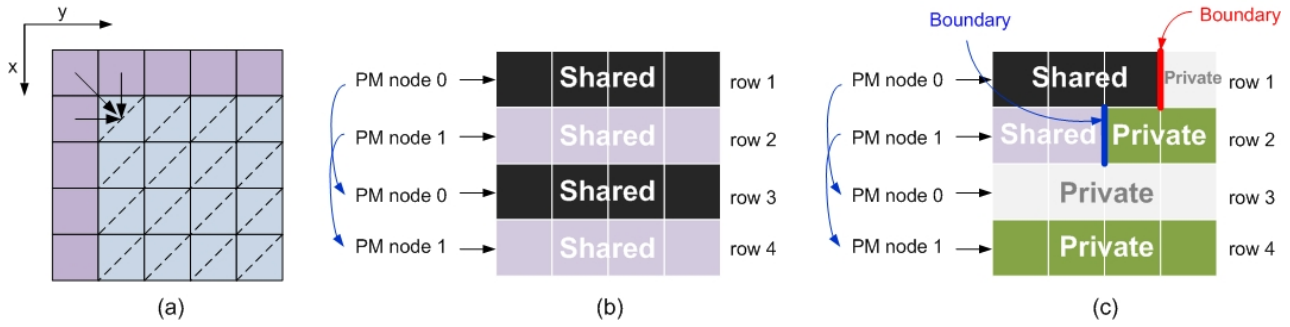


Figure 11. (a) Memory allocation for Wavefront Computation, (b) conventional DSM organization, and (c) Hybrid DSM organization with run-time partitioning

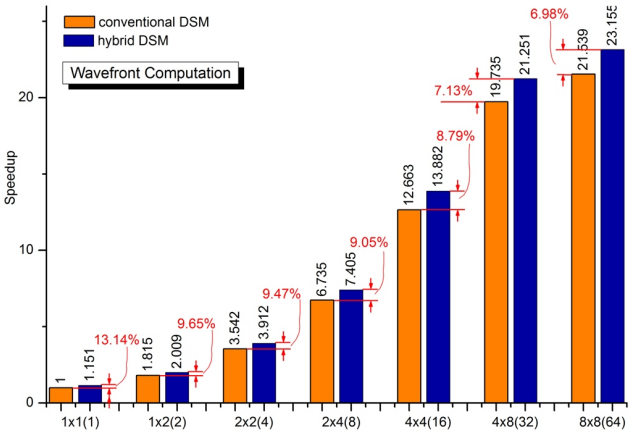


Figure 12. Speedup and performance improvement of Wavefront Computation

illustrates, the elements, which haven't been computed by their host PM node yet, are *private*. The host PM node accesses them in fast *physical addressing scheme*. After the host PM node accomplishes computation on them, it re-configure the boundary address to make them be *shared* so that other PM node could access them. There is a synchronization point after the host PM node re-configure the boundary address. Compared with aforementioned Matrix Multiplication and 2D DIT FFT, Wavefront Computation has a number of boundary address adjustment.

Fig. 12 shows the speedup and performance improvement of the Wavefront Computation. As the system size increases from 1 to 2, 4,, 8, 16, 32, and 64, the speedup with conventional DSM organization goes up from 1 to 1.815, 3.542, 6.735, 12.663, 19.735 and 21.539 and the speedup with hybrid DSM organization is from 1.151 to 2.009, 3.912, 7.405, 13.882, 21.251 and 23.155. Since synchronization-intensive, the speedup of Wavefront Computation is lower than that of Matrix Multiplication and 2D DIT FFT, especially as the system size becomes larger. To calculate the performance gain induced by hybrid DSM organization, the speedup for hybrid DSM organization is normalized with the execution time of single PM node in conventional DSM organization. As is shown in Fig. 12 , for the different system sizes of 1, 2, 4, 8, 16, 32 and 64, the performance improvement is 13.14%, 9.65%, 9.47%, 9.05%, 8.79%, 7.13% and 6.98%, respectively.

From Fig. 12, we can see that

- In Wavefront Computation algorithm, all PM nodes act as both a producer and a consumer. As shown in Fig. 11 (b) and (c), PM node 0 acts as a producer who computes the elements in row 1. PM node 1 acts as a consumer who uses the elements in row 1. Meanwhile PM node 1 also a producer who computes the elements in row 2 which are used by PM node 0 when it computes the elements in row 3. It fits the *producer-consumer mode* in Subsection IV-B. Using *producer-consumer mode* of run-time partitioning in hybrid DSM organization, we could reduce and hide V2P address translation overhead by fast *physical addressing scheme* and concurrent memory addressing, respectively.
- In conventional DSM organization, the obtained speedup is 21.539 for 8×8, a little higher than 19.735 for 4×8. It's the same with hybrid DSM organization. This is because synchronization overhead (it's mainly waiting time) and network communication delay become dominating as the system size is scaled up gradually and degrade the performance.
- The single PM node case has a higher improvement because all data accesses are local shared for the conventional DSM organization and private for the hybrid DSM organization, and there is no synchronization overhead.

VI. Concluding Remark

DSM organization offers ease of programming by maintaining a global virtual memory space as well as imports the inherent overhead of Virtual-to-Physical (V2P) address translation, which leads to negative performance. Observing that it's unnecessary to perform V2P address translation for private data accesses, this paper introduces hybrid DSM organization and run-time partitioning technique in order to improve the system performance by reducing V2P address translation overhead as much as possible. The philosophy of our hybrid DSM organization is to support fast and physical memory accesses for private data as well as to maintain a global and single virtual memory space for shared data. The run-time partitioning supports re-configuration of the hybrid DSM organization

by dynamically changing the boundary address of private memory region and shared memory region during the program execution. The experimental results of real applications show that the hybrid DSM organization with runtime partitioning demonstrates performance advantage over the conventional DSM counterpart. The percentage of performance improvement depends on problem size, way of data partitioning and computation/communication ratio of parallel applications, system size, etc. In our experiments, the maximal improvement is 34.42%, the minimal improvement 3.68%. In the future, we shall extend our work to cover more applications.

REFERENCES

[1] M. Horowitz and W. Dally, "How scaling will change processor architecture," in *Int'l Solid-State Circuits Conf. (ISSCC'04), Digest of Technical Papers*, 2004, pp. 132–133.

[2] S. Borkar, "Thousand core chips: A technology perspective," in *Proc. of the 44th Design Automation Conf. (DAC'07)*, 2007, pp. 746–749.

[3] A. Jantsch and H. Tenhunen, *Networks on chip*. Kluwer Academic Publishers, 2003.

[4] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Comp. Surveys*, vol. 38, no. 1, pp. 1–51, Mar. 2006.

[5] J. D. Owens, W. J. Dally, *et al.*, "Research challenges for on-chip interconnection networks," *IEEE MICRO*, vol. 27, no. 5, pp. 96–108, Oct. 2007.

[6] S. Vangal, J. Howard, G. Ruhl, *et al.*, "An 80-tile 1.28tflops network-on-chip in 65nm cmos," in *Int'l Solid-State Circuits Conf. (ISSCC'07), Digest of Technical Papers*, 2007, pp. 98–100.

[7] E. Marinissen, B. Prince, D. Keltel-Schulz, and Y. Zorian, "Challenges in embedded memory design and test," in *Proc. of Design, Automation and Test in Europe Conf. (DATE'05)*, 2005, pp. 722–727.

[8] X. Chen, Z. Lu, A. Jantsch, and S. Chen, "Supporting distributed shared memory on multi-core network-on-chips using a dual microcoded controller," in *Proc. of the Conf. on Design, automation and test in Europe (DATE'10)*, 2010, pp. 39–44.

[9] M. Monchiero, G. Palermo, C. Silvano, and O. Villa, "Exploration of distributed shared memory architecture for noc-based multiprocessors," in *Proc. of the 2006 Int'l Conf. on Embedded Computer Systems: Architectures, Modeling and Simulation*, 2006, pp. 144–151.

[10] L. Xue, O. Ozturk, F. Li, M. Kandemir, and I. Kolcu, "Dynamic partitioning of processing and memory resources in embedded mpsoc architectures," in *Proc. of the Conf. on Design, automation and test in Europe (DATE'06)*, 2006, pp. 690–695.

[11] S. Srinivasan, F. Angiolini, M. Ruggiero, L. Benini, and N. Vijaykrishnan, "Simultaneous memory and bus partitioning for soc architectures," in *Proc. of IEEE Int'l Conf. on SoC (SoCC'05)*, 2005, pp. 125–128.

[12] S. Mai, C. Zhang, and Z. Wang, "Function-based memory partitioning on low power digital signal processor for cochlear implants," in *Proc. of IEEE Asia Pacific Conf. on Circuits and Systems (APCCAS'08)*, 2008, pp. 654–657.

[13] A. Macii, E. Macii, and M. Poncino, "Improving the efficiency of memory partitioning by address clustering," in *Proc. of the Conf. on Design, automation and test in Europe (DATE'03)*, 2003, pp. 18–23.

[14] G. Suh, L. Rudolph, and S. Devadas, "Dynamic partitioning of shared cache memory," *J. Supercomputing*, vol. 28, no. 1, pp. 7–26, Apr. 2004.

[15] X. Qiu and M. Dubois, "Moving address translation closer to memory in distributed shared-memory multiprocessors," *IEEE trans. Parallel and Distributed Systems*, vol. 16, no. 7, pp. 612–623, 2005.

[16] P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design tradeoffs for network-on-chip interconnect architectures," *IEEE Trans. on Computers*, vol. 54, no. 8, pp. 1025–1040, Aug. 2005.

[17] "Leon3 processor," in *http://www.gaisler.com*.

[18] A. Jantsch *et al.*, "The nostrum network-on-chip," in *http://www.ict.kth.se/nostrum*.

[19] W. Zhu, V. Sreedhar, Z. Hu, and G. Gao, "Synchronization state buffer: supporting efficient fine-grain synchronization on many-core architectures," in *Proc. of the 34th annual Int'l Symp. on Computer Architecture (ISCA'07)*, 2007, pp. 35–45.

Xiaowen Chen received two B.S. degrees in Microelectronics and Computer Science, respectively, from the University of Electronic Science and Technology of China (UESTC), in 2005. He is now a joint PhD candidate majoring Microelectronics both in the National University of Defense Technology (NUDT), Changsha, China, and KTH-Royal Institute of Technology, Stockholm, Sweden. His research interests include computer architecture, microarchitecture, VLSI design, system-on-chips, network-on-chips, Distributed Shared Memory.

Shuming Chen received BSc., MSc. and PhD. from National University of Defense Technology (NUDT), Changsha, China, in 1982, 1988 and 1993, respectively. Since then he is with School of Computer, National University of Defense Technology (NUDT). Currently, he is a full professor in microprocessor design. His research interests include processor architecture, high performance circuits, custom design for reliability, and SOC. S. Chen has published over 110 papers in conferences and journals in the area of microarchitecture, VLSI design, and Digital Signal processor. As a chief architect, he designed more than ten microprocessor chips in recent years.

Zhonghai Lu received BSc. from Beijing Normal University, China in 1989. Since then he had worked extensively in industry for several electronic, communication and embedded systems companies as a system engineer and project manager for eleven years. Afterwards, he entered the Royal Institute of Technology (KTH), Sweden in 2000. From KTH, he received MSc. and PhD. in 2002 and 2007, respectively. He is currently a researcher at KTH. Dr. Lu has published over 25 peer-reviewed technical papers in journals, book chapters and international conferences in the areas of networks/systems on chips, embedded real-time systems and communication networks. His research interests include computer systems and VLSI architectures, interconnection networks, system-level design and HW/SW co-design, reconfigurable and parallel computing, system modeling, refinement and synthesis, and design automation.

Axel Jantsch received a Dipl.Ing. (1988) and a Dr. Tech. (1992) degree from the Technical University Vienna. Between 1993 and 1995 he received the Alfred Schrdinger scholarship from the Austrian Science Foundation as a guest researcher at the Royal Institute of Technology (KTH). From 1995 through 1997 he was with Siemens Austria in Vienna as a system validation engineer. Since 1997 he is with the Royal Institute of Technology, Stockholm, Sweden. Since December 2002 he is full professor in Electronic System Design. A. Jantsch has published over 140 papers in international conferences and journals in the areas of VLSI design and synthesis, system level specification, modeling and validation, HW/ SW co-design and co-synthesis, reconfigurable computing and networks on chip. At the Royal Institute of Technology A. Jantsch is heading a number of research projects, in the areas of system level specification, design, synthesis, validation and networks on chip.

Design and Evaluation of an Online Anomaly Detector for Distributed Storage Systems

Xin Chen[1], Xubin He[2], He Guo[3] and Yuxin Wang[4]
[1]Department of Electrical and Computer Engineering, Tennessee Technological University, Cookeville, TN, USA
[2]Department of Electrical and Computer Engineering, Virginia Commonwealth University, Richmond, VA, USA
[3]School of Software Technology, Dalian University of Technology, Dalian, LiaoNing, China
[4]School of Computer Science and Technology, Dalian University of Technology, Dalian, LiaoNing, China
Email: xchen21@students.tntech.edu, xhe2@vcu.edu, {guohe, wyx}@dlut.edu.cn

***Abstract*— Performance problems, which may stem from different system components, such as network, memory, and storage devices, are difficult to diagnose and isolate in distributed storage systems. In this paper, we present a performance anomaly detector which is able to efficiently detect performance anomaly and accurately identify the faulty sources in a system node of a distributed storage system. Our method exploits the stable relationship between workloads and system resource statistics to detect the performance anomaly and identify faulty sources which cause the performance anomaly in the system. Our experimental results demonstrate the efficiency and accuracy of the proposed performance anomaly detector.**

***Index Terms*— performance anomaly detector, distributed storage systems, parallel file systems**

I. Introduction

Performance is critical in the study of distributed storage systems. Synthetic workloads or file system benchmarks are created to examine the behaviors of storage systems. Although they are very useful in the initial stage of the design and development of storage systems, it is insufficient for using them to analyze or resolve one common problem called performance anomaly in these systems [1], [2]. By performance anomaly it means that the observed system behaviors are not expected according to the observed system workload. For example, I/O throughput has a significant degradation given a moderate amount of I/O requests. Performance anomaly is closely related to either some resource-intensive processes that demand large portion of system resources (CPU or memory) or some unexpected software and hardware behaviors like software bugs (memory leaking) and hardware faults (bad hard drive sectors), and it is common in storage systems. However, it remains a challenging task to efficiently detect performance anomaly and accurately identify the faulty sources, particularly in distributed storage systems.

Distributed storage systems usually consist of a large amount of commodity computer nodes which may have different processing capabilities. However, the overall performance of such systems is not determined by the fastest computer nodes of the systems, instead, the performance is often limited by the capability of the slowest ones [2], [3]. So, if there exists performance anomaly in some node of a distributed storage system, it is highly possible that the overall system performance will suffer negative effects, and such effects may be accumulated and magnified due to long-running and large-scale computations [2], which directly hurts the reliability and availability of the system. Therefore, it is necessary and crucial to equip distributed storage systems with a tool which is able to efficiently detect performance anomaly and accurately identify the faulty sources.

As compared to the fail-stop failures [4], it is more difficult to detect the existence of performance anomaly, and even more difficult to identify the source of the anomaly, because both dynamic workload change and many uncertain factors such as caching and scheduling can perplex people's ability to understand the system behaviors. Currently, some anomaly detecting approaches are threshold-based, which set thresholds for observed system metrics and raise signals when the thresholds are violated [5], [6]. However, it is difficult to choose appropriate thresholds for a variety of workloads and computer nodes with different capabilities. Some approaches are model-based, which indicate performance anomaly by comparing the observed system measurements and the model estimations [2], [7], [8], however, their usages are limited to the generality of the models.

This work targets the runtime diagnosis of performance anomaly in distributed storage systems which may consist of heterogeneous computer nodes and experience dynamic changed workloads. The proposed approach is self-diagnosis based, which exploits some invariants that exist in a computer node of a distributed storage system to detect the performance anomaly and identify faulty sources of that node. Such invariants refer to the stable relations between workloads and system resource statistics in faulty-free situations.

The rest of the paper is organized as follows: Section II gives a brief discussion on related work. In section III, we describe our methodology for performance anomaly detection and identification, and present the design of our performance anomaly detector in section IV. Section V describes our experiments and lists experimental results. Finally, we conclude the paper in section VI.

doi:10.4304/jsw.6.12.2379-2390

II. Related Work

For large-scale systems like cluster file systems, it is a major challenge to understand system behaviors, particularly unexpected behaviors. Numerous techniques have been proposed for detecting system anomalies. Among them, the simplest ones are the threshold-based techniques which are a form of service level agreements (SLAs). They are very useful on the condition that their users clearly know the key metric to monitor and the best value of the thresholds in different scenarios [5], [6]. Unfortunately, it is very difficult, even for an expert, to correctly choose the necessary metrics to monitor and set the right values of the thresholds for different scenarios in the context of today's complex and dynamic computer systems.

Recently, statistical learning or data mining techniques are widely employed to construct probability models for detecting various anomalies in large-scale systems based on some heuristics and assumptions, although these heuristics and assumptions may only hold in some particular systems or scenarios.

Kasick et al [2] developed a statistical peer-comparison diagnosis approach to identify a faulty node in a cluster file system. The rationale of their approach is based on the observation that there is an obvious difference between the behaviors of fault-free and faulty nodes. Kavulya et al [9] and Lan et al [8] proposed the similar approaches to detect performance problems in replicated file systems and a cluster system, respectively. However, the validation of these approaches is based on a strong assumption of homogeneous hardware and workloads, which may only hold in a few cases.

Besides the probability models for system metrics such as throughput, response time, etc, various relationships and correlations among system inputs and measurements are also explored and modeled to detect anomalies in large-scale computer systems. Chen et al [7] developed a new technique, the principal canonical correlation analysis (PCCA), to perform failure detection in large-scale computer systems which provide online Internet services. The key idea of their approach is to capture the contextual relationships between the system inputs and their internal measurements which hold in fault-free scenarios, and are broken in faulty scenarios. However, it is required for applying their technique that there exists a linear relationship between the system inputs and their internal measurements.

Guo et al [10] and Gao et al [11] investigated the probabilistic correlation between flow-intensities measured at different points and the one between different system measurements, respectively. In this work, we also exploit the correlation among system measurements, however, we not only use them to detect the existence of performance anomaly in a cluster file system, but also pinpoint the source of the performance anomaly.

III. Performance Anomaly Detection and Identification

Given the scale and heterogeneity of distributed storage systems, it is usually difficult to perform peer comparison to distinguish faulty nodes from fault-free nodes, because the behaviors of nodes in these systems may not be comparable. In this work, a self-diagnosis based approach is adopted to detect the existence of performance anomaly and identify the faulty resources. The major advantage of the approach is its independence of the scale and heterogeneity of distributed storage systems.

The feasibility of the approach is based on two observations. First, resource overuse (CPU and memory) and hard disk faults are very common in today's distributed storage systems according to the recent studies [12]–[19]. They manifest themselves at least on a computer node. Thus, if it is able to identify the system abnormal behaviors originated from them by analyzing system measurements collected on a computer node, it is not necessary to adopt centralized or peer comparison based performance anomaly detectors, which are expensive and not practical for heterogeneous computer systems.

Second, there exist some relations among the system measurements of a computer node in distributed storage systems, which can be regarded as invariants when the node works properly; but, one or more of such invariants does not hold once the system experiences performance anomaly [10], [20]. Such observation lays a strong foundation for performing self-diagnosis based performance anomaly detection and faulty resource identification, because performance anomaly can be detected and faulty sources can be identified by simply checking whether some invariants hold or not. Therefore, the main task is to figure out the invariants of distributed storage systems which can work as an indicator of performance anomaly.

A. Relation among Computer Nodes in Distributed Storage Systems

Before exploring the invariants of distributed storage systems, it is necessary to understand the relation among computer nodes in these systems. These distributed storage systems typically consist of three main components: clients, a metadata server or a server cluster (MDS), and a cluster of I/O servers or object storage devices (OSDs). They provide an inexpensive alternative utilizing Commodity Off The Shelf (COTS) products allowing large I/O intensive applications to be run on high performance clusters [21]. Figure 1 presents a general architecture of such systems. In this architecture, metadata operations are separated from I/O operations, and there exist two types of relations among system nodes: the relation between a metadata server and multiple I/O servers and the relation among a set of I/O servers.

The first type of relation reveals a single point of failure in these storage systems. Because metadata servers are always accessed before actual data transferring, once metadata servers are down, clients cannot initiate any I/O operations. On the other side, metadata severs do not

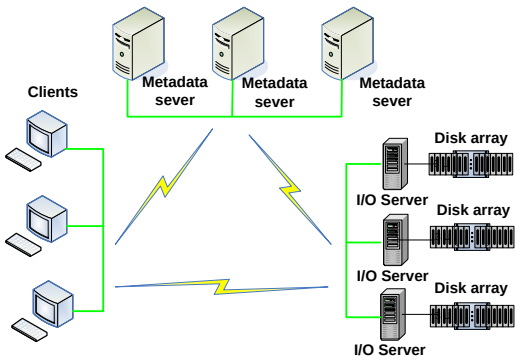


Figure 1: The general architecture of distributed storage systems.

intervene the actual I/O processing between clients and I/O servers. Even if a metadata sever is down during an I/O operation, the I/O operation can still be completed [22]. In this work, because the concentration is I/O performance of a distributed storage system, the relation between a metadata server and multiple I/O servers is not considered. The main focus is on studying the relation among a set of I/O servers which is more relevant to system I/O performance.

In a distributed storage system which deploys file systems like PVFS [23] and Lustre [24], data are usually distributed among a set of I/O servers. An I/O request from a client normally consists of several sub-I/O requests corresponding to different I/O servers. When a client processes such an I/O request, it is blocked on some syscall until it properly receives all I/O server responses, and then it can process next I/O request. Figure 2 presents an example of an I/O request sequence from a client. In this example, there are total three I/O requests, and each I/O request contains three sub-I/O requests. For example, sub-I/O request 1, 2, and 3 are of the first I/O request from the client, and they are issued to I/O server 1, 2, and 3, respectively. Let s_1, s_2, and s_3 denote the request sending rate of the client to I/O server 1, 2, and 3 respectively, and they are calculated by Formula 1.

$$\begin{cases} s_1 = \dfrac{\text{I/O request 1 + I/O request 4 + I/O request 7}}{t_3 - t_0} \\ s_2 = \dfrac{\text{I/O request 2 + I/O request 5 + I/O request 8}}{t_3 - t_0} \\ s_3 = \dfrac{\text{I/O request 3 + I/O request 6 + I/O request 9}}{t_3 - t_0} \end{cases} \quad (1)$$

Thus, when an I/O server experiences a performance problem, it definitely increases the client's response time

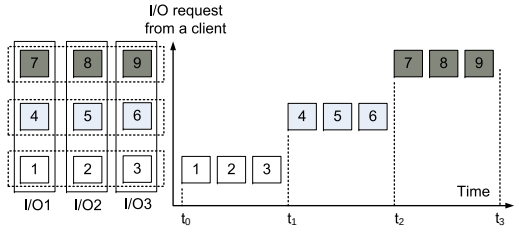


Figure 2: An I/O request sequence from a client.

(service time + waiting time), and as a result, s_1, s_2, and s_3 will decrease due to the increase of $t_3 - t_0$. Because the request receiving rate of an I/O server is proportional to the sending rate of a client, an important relation among a set of I/O servers can be concluded that once an I/O server experiences a performance problem (e.g., I/O performance decrease), it receives fewer requests per time unit from clients, so do other I/O servers.

Figure 3 shows I/O request receiving rates of a normal node and faulty node in the presence of a performance anomaly. It is clear that when a performance problem occurs at a computer node, the problem not only reduces the amount of received request per second at the faulty node, but also manifests similar symptom at other nodes which work properly.

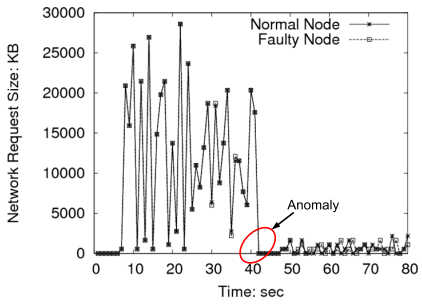


Figure 3: I/O request receiving rate during a sequential write. At the 41st second, disk delay faults were injected in a computer node, which produced a performance anomaly.

B. *Invariants*

Various relations among system measurements exist in a computer node of a distributed storage system. Here, invariants refer to those stable relations when a system properly works. Because I/O performance is very important in distributed storage systems, in this work, the focus is on how to discover and utilize invariants in a computer node to detect and pinpoint I/O performance problems.

Because any performance problem at a computer node manifest symptoms of unexpected certain resource usage, because system resources are always limited, once one or more processes occupies too many resources and does not release them, the executions of other processes are negatively impacted, as the OS kernel forces the processes sleep until the required resources are ready [25]. Meanwhile, if a resource request from a process cannot be satisfied immediately, the kernel also forces the process sleep. Thus, one option of utilizing invariants in a computer node to detect performance anomaly is to explore the relations between workloads and system resource statistics.

To facilitate the discussion, how a computer node handles the I/O requests from clients is first studied. Figure 4 depicts an I/O request flow in a computer node. External I/O requests are first processed by the process

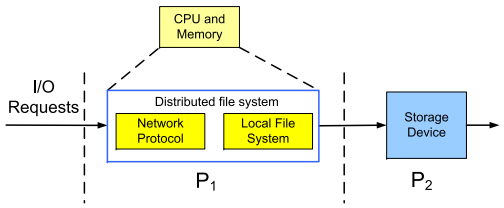


Figure 4: An I/O request flows in a computer node.

of a distributed file system on the node, then the process sends the I/O requests to a hard disk, and the hard disk finally completes those I/O requests. According to the location of I/O requests, the flow can be divided into two phases: P_1 and P_2. The former indicates the phase where I/O requests are processed by the distributed file system, and the latter represents the phase where I/O requests are in a hard disk.

In the two phases, different resources are required for processing incoming I/O requests. CPU and memory are two major required resources in P_1, as in that phase, a distributed file system transforms the I/O requests from network into the ones for the local disk. I/O requests are finally stratified in P_2, the local disk is the major required resource in that phase. Thus, the concentration is of analyzing the relations between workloads and the statistics of the resources listed above to look for the invariants which can be used to detect and pinpoint performance problems. By studying the trace data collected from the previous studies on distributed storage systems [3], [22], three invariants are concluded as follows based on the statistics listed in Table I.

Invariant for memory. If the process of a distributed file system at a computer node works properly, without intervention of other processes, the total size of I/O requests over network per second is proportional to the amount of the allocated memory per second.

Memory is allocated to hold data either after the arrival of write requests from clients or before sending back the satisfied read requests to clients. Thus, if a computer node has sufficient free memory and there are no other memory intensive processes running on the node, the total size of I/O requests over network per second is proportional to the amount of the allocated memory per second. The invariant is used to identify the performance problems originated from memory. Figure 5 gives an example of the invariant.

Invariant for CPU. If the process of a distributed file system at a computer node works properly, without intervention of other processes, the total size of I/O requests over network per second is proportional to the number of interrupts per second.

Interrupts are generated during the processing of I/O requests. For example, a network interface card raises hardware interrupt to CPU after the arrival of I/O requests from clients; disk interrupts are triggered when I/O requests are issued to a hard disk drive. If more I/O requests arrive at a computer node, more interrupts are generated, and vice versa. Meanwhile, the generation rate

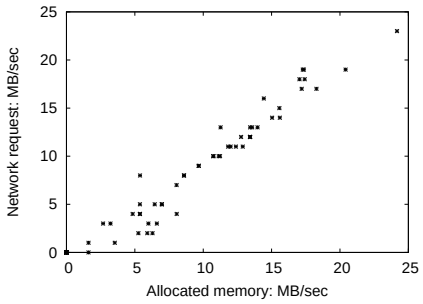


Figure 5: The relation between the total size of I/O requests over network per second and the amount of allocated memory per second. Data is from a trace of 40 seconds I/O activities in a computer node.

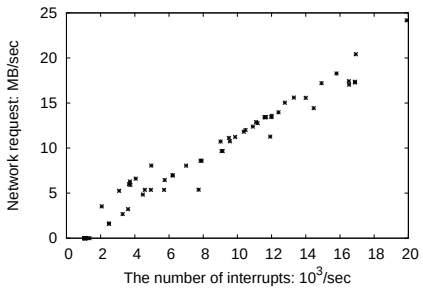


Figure 6: The relation between the total size of I/O requests over network per second and the number of interrupts per second. Data is from a trace of 40 seconds I/O activities in a computer node.

of interrupts is closely related to the CPU time of the corresponding process, as it requires a significant amount of CPU time to process I/O related interrupts [26]. Once the CPU resource is insufficient for the distributed file system process, fewer I/O related interrupts are generated, and the proportional relation between I/O request arrival rate and interrupt generating rate does not hold. The invariant is used to identify the performance problems originated from CPU. Figure 6 gives an example of the invariant.

Invariant for disks. If a hard disk works properly and has continued I/O requests, the average I/O request size is proportional to the average I/O request service time.

I/O requests issued to a hard disk usually have different sizes. It is intuitive that larger requests require more service time than smaller requests. However, when hard disks process discontinued I/O requests, small requests may require more service time than large requests, because the disk seek time dominates the total request service time. Thus, when a hard disk works properly and has continued I/O requests, the average I/O request size is proportional to the average I/O request service time. The invariant is used to identify the performance problems originated from hard disks. Figure 7 gives an example of the invariant. In the figure, the proportional relation is maintained among I/O requests with large size, but if I/O request size is very

TABLE I.: Workload and System Resource Statistics.

Source	Metric	Description
Workloads	net_req	the total size of I/O requests over network per second.
	avgrq-sz	The average size (in sectors) of the requests that are issued to a disk drive.
System resource	free_mem	The amount of idle memory.
	in	The number of interrupts per second, including the clock.
	svctm	The average service time (in milliseconds) for I/O requests that are issued to a disk drive.

small, the proportional relation rarely holds.

C. Indicators

According to the relation among I/O servers concluded in Section III-A, once an I/O server has a performance problem, the problem will be also observed at other I/O servers. Such a relation makes it difficult to accurately locate the faulty server. Furthermore, dynamically changing workloads perplex people's ability to determine appropriate thresholds to identify performance anomaly [1]. It is necessary to find indicators which are highly sensitive to performance anomalies but less sensitive to other factors. In this work, the invariants discussed in the previous section are leveraged to develop such indicators.

1) Indicators of Performance Anomalies: Although one or more of the above invariants does not hold when an I/O server experiences performance problem, it is still insufficient to only depend on them to detect the existence of performance anomaly on the server, because even if when an I/O server works properly, these invariants may still not hold, for example, marginal memory allocation by other processes may break the invariant for memory but does not negatively impact the running of the process of a distributed file system on the server.

To compensate the drawback of the invariants, an indicator $I_{req}(n)$ is adopted to detect the performance anomaly on an I/O server at the nth sampling period. Formula 2 gives the definition of $I_{req}(n)$, where req_{n-1} denotes the average total size of I/O requests at the $(n-1)$th sampling period, req_n denotes the average total size of I/O requests at the nth sampling period, and α denotes a threshold of the degradation ratio between req_{n-1} and req_n. If the ratio is greater than or equal to α, $I_{req}(n)$ generates a TRUE value, which suggests a performance problem, otherwise not. Similarly, $I_{req}(n)$ cannot be used alone to detect performance anomaly on an I/O server, because non-faulty I/O servers also observer the degradation of receiving request rate. $I_{req}(n)$ should be combined with the indicators of the invariants to detect performance anomaly.

$$I_{req}(n) = \begin{cases} \text{FALSE, if } \frac{req_{n-1}-req_n}{req_n} < \alpha, req_n \neq 0 \\ \text{FALSE, } req_n = 0 \\ \text{TRUE, if } \frac{req_{n-1}-req_n}{req_n} \geq \alpha, req_n \neq 0 \end{cases} \tag{2}$$

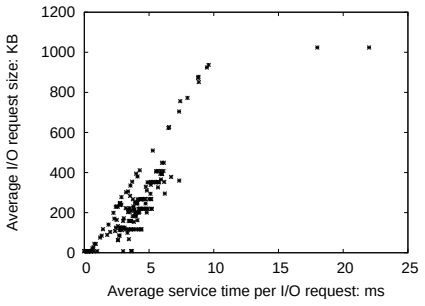


Figure 7: The relation between average I/O request size and average service time per I/O request. Data is from a trace of 250 seconds I/O activities on a computer node.

2) Indicators of Faulty Sources: Because the invariants discussed above refer to a proportional relation between two metrics, in order to use them in practice, such a proportional relation needs to be quantified. The correlation $corr(x, y)$ is a good measurement for quantifying a proportional relation between two variables: x and y. Formula 3 gives a formal definition of $corr(x, y)$, where $\sigma_{x,y}$ denotes the covariance of x and y; σ_x and σ_y denote the variance of x and y, respectively; μ_x and μ_y represent the mean value of x and y, respectively; $E(x)$ calculates the expectation of variable x. The sign of $corr(x, y)$ is more meaningful than its absolute value: once correlation is positive, it indicates x increases as the increase of y; otherwise, it indicates x is not proportional to y.

$$corr(x,y) = \frac{\sigma_{x,y}}{\sigma_x \sigma_y} = \frac{E[(x-\mu_x)(y-\mu_y)]}{\sqrt{E(x-\mu_x)^2}\sqrt{E(y-\mu_y)^2}} \tag{3}$$

Thus, based on Formula 3, three indicators I_{mem}, I_{cpu}, and I_{disk} are defined to test the invariants by Formula 4, 5, and 6, respectively. If an indicator has a boolean value of TRUE, the corresponding invariant holds, otherwise, the invariant does not hold, which suggests the performance problem originates from the corresponding resource. Table II lists the parameters used in these formulas.

TABLE II.: Symbols in Formulas 4, 5, and 6.

Parameter	Description
req	the total size of incoming I/O requests per second.
$interrupt$	the number of generated interrupts per second.
mem	the amount of allocated memory per second.
$iosize$	the average I/O request size to a hard disk per second.
$svctm$	the average I/O request service time per second.

TABLE III.: An Example of a Probability Distribution Table of $P(iosize|svctm)$. Data is from a trace of 250 seconds I/O activities in a computer node.

I/O request size: KB	Average service time: ms		
	[0, 8)	[8, 12)	[12,)
[0, 150)	100%	0	0
[150,)	85%	10%	4%

$$I_{cpu} = \begin{cases} \text{FALSE, if } corr(req, interrupt) < 0 \\ \text{TRUE, if } corr(req, interrupt) \geq 0 \end{cases} \quad (4)$$

$$I_{mem} = \begin{cases} \text{FALSE, if } corr(req, mem) < 0 \\ \text{TRUE, if } corr(req, mem) \geq 0 \end{cases} \quad (5)$$

$$I_{disk} = \begin{cases} \text{FALSE, if } corr(iosize, svctm) < 0 \\ \text{TRUE, if } corr(iosize, svctm) \geq 0 \end{cases} \quad (6)$$

Because I_{disk} does not work well in the case of discontinued I/O requests, in order to compensate the drawback of I_{disk}, the conditional probability distribution $P(iosize|svctm)$ is checked to confirm the value generated by I_{disk}; as if a hard disk works stable, a stable probability distribution has to be observed. Table III gives an example of the probability distribution. For example, if a sequence of {(80KB, 3ms), (81KB, 2ms), (82KB, 4ms), (83KB, 2ms)} is observed, although I_{disk} generates a FALSE value, the $P(iosize|svctm)$s of all pairs in the sequence are 100% according to Table III, thus, the FALSE value of I_{disk} is not confirmed and a TRUE value is generated. In this work, once I_{disk} generates a FALSE value, and the corresponding $P(iosize|svctm)$ is less than 10%, the FALSE value of I_{disk} is confirmed, otherwise, I_{disk} generates a TRUE value.

Although higher accuracy can be achieved by checking $P(iosize|svctm)$ for identifying disk problems than only looking at the proportional relation between $iosize$ and $svctm$, the major drawback of the method is the long training time which is required for collecting sufficient data to calculate a dependable probability distribution. So, I_{disk} serves as the major indicator of disk problems in the absence of a dependable $P(iosize|svctm)$.

As compared to the performance problems originated from memory, CPU, and hard disks, the problems from network are more difficult to diagnose, as they usually manifest themselves as a symptom of workload change, and it is difficult to only use the local information of an I/O server to identify them. An indicator $I_{network}$ is defined by Formula 7, which combines the local information of an I/O server and the information from other related I/O servers to identify the network problems. In Formula 7, $I^n_{network}$ is a local indicator of network on an I/O server n, its TRUE value suggests there may have some network problem which causes the performance anomaly, but the value should be confirmed by the external information from other I/O servers; $I'_{network}$ finally determines whether the network is a faulty source or not, if a TRUE value is generated by it, the source of performance anomaly can be pinpointed to the network.

$$\begin{cases} I^n_{network} = I^n_{disk} \wedge I^n_{mem} \wedge I^n_{cpu} \wedge I^n_{req},\ n \in N \\ I'_{network} = I^1_{network} \wedge I^2_{network} \wedge \cdots \wedge I^n_{network},\ n \in N \end{cases} \quad (7)$$

IV. The Design of the Online Performance Anomaly Detector

The online performance anomaly detector is implemented as a daemon process which runs at each computer node of a cluster file system. The detector sends alarms to clients or administration nodes, when performance anomaly is detected at a computer node. It is worth pointing out that once performance anomaly is detected on a computer node, it is most likely that the other computer nodes generate alarms soon, and those alarms may mark other resource as faulty, meanwhile, one or more of our invariants on the computer node may not hold any more until the performance anomaly is fixed. Thus, the alarms raised after the first alarm in a short period are ignored.

Figure 8 shows the working flow of our performance anomaly detector. The detection process is triggered when there is a significant degradation of req, then all indicators are evaluated accordingly to identify which system component is the faulty source, finally an alarm is raised if the performance anomaly is detected.

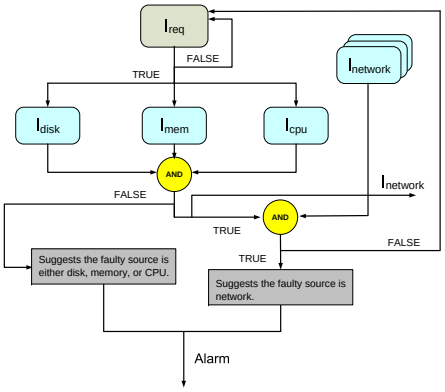


Figure 8: The working flow of the online performance anomaly detector.

V. Experiments

To demonstrate the efficiency of our performance detector, we constructed a testbed which consisted of four computer nodes (1 metadata server, 3 I/O servers). These servers have different computation and I/O capabilities, as shown in table IV. Our detector was evaluated with synthetic workloads on a parallel file system, PVFS. Four faults were injected to produce faulty situation during the evaluation: disk delay faults, network delay faults, CPU overuse faults, and memory overuse faults. disk delay faults introduce extra I/O request processing time in a hard disk driver; network delay faults add extra delay at an I/O server for sending every request over the network; CPU and memory overuse faults limit the available CPU

and memory resource at a low level, respectively. In our experiments, we adopted a sampling period of four seconds according to our prior experience, in which four samples were taken, one per second, and all indicators were evaluated at the end of the period; we set α to 50% for I_{req}.

In order to measure the efficiency and accuracy of our detector, two metrics are defined: the detection latency and the true positive rate. The former measures how long our detector may take to detect the existence of performance anomaly after the injection of performance faults, and the latter measures the accuracy of our detector in terms of the percentage of correct alarms. Formula 8 and 9 give the definitions of the two metrics, where Δ denotes the detection latency, T_d represents the time point at which performance anomaly is detected, T_i denotes the fault injection time point, A_{td} denotes the true positive rate, N_{td} and N_{fd} represent the number of true and false detections, respectively.

$$\Delta = T_d - T_i \tag{8}$$

$$A_{td} = \frac{N_{td}}{N_{td} + N_{fd}} \tag{9}$$

In this section, the behaviors of our performance anomaly detector are examined with synthetic workloads in different faulty situations. Before the discussion of our detector in faulty situations, the system behaviors in fault-free situation are studied first; we focus on examining whether our invariants hold or not, which is of ultra importance for the correctness of our detector.

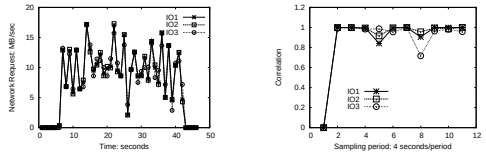
Figure 9 shows the results of 1GB sequential write tests on PVFS in fault-free situation. In the these figures, our three invariants perfectly hold in the presence of a significant fluctuations of external I/O request rate for both file systems, as the values of three correlations along the time axis are almost positive. The only exception is in figure 9d, the correlation of $iosize_n$ and $svctm_n$ is negative at the second sampling period. However, it is reasonable, as in the period, I/O servers just started to process I/O requests, hard disks may take relative long service time for processing the first incoming I/O requests with moderate sizes, which breaks the third invariant.

The results of 1GB sequential read tests on PVFS in fault-free situation are shown in figure 10. As similar as in figure 9, the invariants for memory and CPU hold through the tests, but the invariant for disk does not always hold, as there is no data caching for PVFS, which results in discontinuous I/O requests. Because there is no significant drop of req in figure 10, even if the invariant is broken, no alarm is raised by our detector in practice.

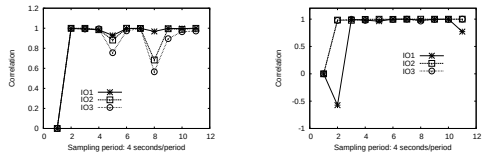
Due to the space limit, we only discussed the results of write tests of the following experiments, and gave a summary of both write and read tests in section V-F.

A. Disk delay faults

This set of experiments evaluated our performance anomaly detector in the case of disk delay faults which do not fail any I/O request but introduce extra I/O request

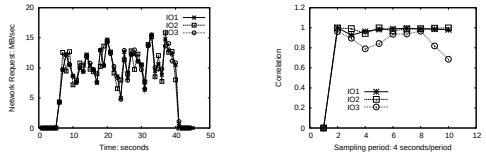


(a) External I/O request rate (b) $corr(req, mem)$

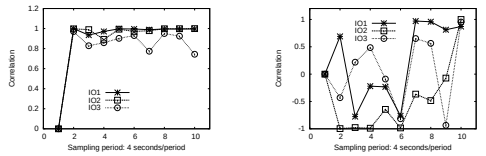


(c) $corr(req, interrupt)$ (d) $corr(iosize, svctm)$

Figure 9: 1GB sequential write on PVFS.



(a) External I/O request rate (b) $corr(req, mem)$



(c) $corr(req, interrupt)$ (d) $corr(iosize, svctm)$

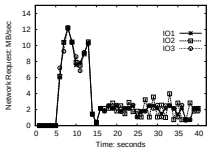
Figure 10: 1GB sequential read on PVFS.

processing time in a hard disk driver. The delay was set to 50 ms for the following experiments. Figure 11 shows the results of 1GB sequential write test on PVFS where the disk delay faults were introduced at the 4th sampling period (13rd – 16th second) at IO2.

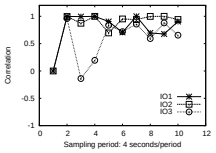
In figure 11, although the invariants for memory and CPU of IO3 do not hold at the 3rd sampling period, req of IO3 does not have a significant drop during such the period which is between the 9th and 12rd second in figure 11a, thus, there was no alarm raised. In the 13rd second, disk delay faults were introduced at IO2, we not only observed a sharp drop of req but also saw the FALSE value generated by I_{disk} of IO2 in the 4th sampling period which includes the 13rd second time point, meanwhile, at the same sampling period, no other invariant was broke. Thus, the performance anomaly was detected, and the faulty source was pinpointed to the hard disk on IO2. Because each indicator generates a boolean value at the end of a sampling period, for this experiment, the latency was $\Delta = 4 \times 4 - 13 = 3$ seconds, and A_{td} was 100%, as there was no false detection.

TABLE IV.: Testbed Information.

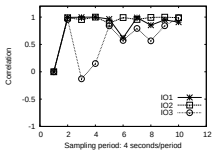
Server name	Type	CPU	Memory	HDD	Network Card
MDS	Metadata server	P4 CPU 2.53GHz	500MB	FUJITSU IDE 8GB 5400rpm	1Gbps
IO1	IO server	P4 CPU 2.40GHz	2026MB	SEAGATE SCSI 18.3GB 15000 rpm	1Gbps
IO2	IO server	P4 CPU 2.40GHz	1264MB	WDC IDE 40GB 7200rpm	1Gbps
IO3	IO server	P4 CPU 2.80GHz	1010MB	WDC SATA 250G 7200rpm	1Gbps



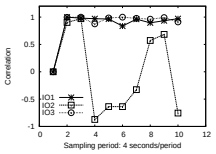
(a) External I/O request rate



(b) $corr(req, mem)$

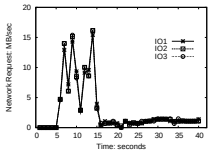


(c) $corr(req, interrupt)$

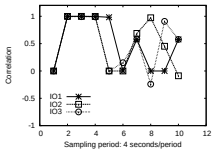


(d) $corr(iosize, svctm)$

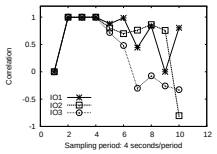
Figure 11: Disk delay faults were injected at the 13rd second, and the workload was 1GB sequential write on PVFS.



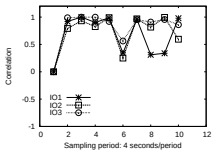
(a) External I/O request rate



(b) $corr(req, mem)$



(c) $corr(req, interrupt)$



(d) $corr(iosize, svctm)$

Figure 12: Network delay faults were injected at 15th second, and the workload was 1GB sequential write on PVFS.

B. Network delay faults

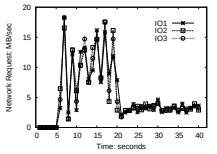
This set of experiments evaluated our performance anomaly detector in the presence of network delay faults which added extra delay at an I/O server for sending every request over the network. The delay was set to 50 ms in the following experiments. Figure 12 shows the results of 1GB sequential write tests on PVFS where the network delay faults were introduced at IO2.

In figure 12, network delay faults were injected at the 15th second at IO2. Our detector correctly detected the performance problem caused by the faults at the 5th sampling period. For this experiment, the detection latency was $\Delta = 5 \times 4 - 15 = 5$ seconds, and the true positive rate was $A_{td} = \frac{4}{4+1} = 0.8$, as the performance anomaly was not detected at the 4th sampling period.

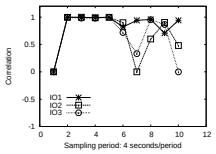
C. CPU overuse

This set of experiments evaluated our performance anomaly detector in the case of CPU overuse faults which make the available CPU resource at a low level. In the set of experiments, our fault injector occupied nearly 90% CPU resource in terms of the percentage of CPU time.

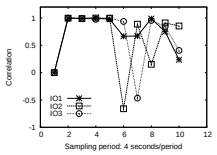
Figure 13 shows the results of the results of 1GB sequential write test on PVFS where CPU overuse faults were injected at the 19th second at IO2. Because I_{req} of IO2 generated a FALSE value at the 5th sampling period, our detector did not raise an alarm. However, our detector raised an alarm at the next sampling period, and correctly pinpointed CPU as the faulty source, as only the invariant for CPU of IO2 was broken. For this experiment, the detection latency was $\Delta = 6 \times 4 - 19 = 5$ seconds, and the true positive rate was $A_{td} = \frac{5}{5+1} \approx 0.83$.



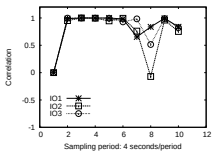
(a) External I/O request rate



(b) $corr(req, mem)$



(c) $corr(req, interrupt)$



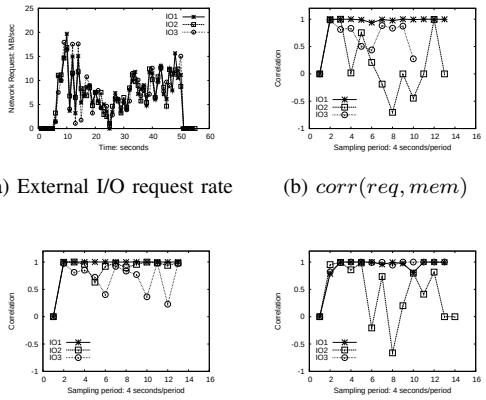
(d) $corr(iosize, svctm)$

Figure 13: CPU overuse faults were injected at 19th second, and the workload was 1GB sequential write on PVFS.

D. Memory overuse

This set of experiments evaluated our performance anomaly detector in the case of memory overuse faults which make the available memory resource at a low level. In the set of experiments, our fault injector occupied up to 90% memory resource.

Figure 14 shows the results of the results of 1GB sequential write test on PVFS where memory overuse faults were injected at the 12nd second at IO2. At the 7th sampling period, I_{req} of IO2 generated a TRUE value, and the invariant for memory of IO2 was broken, an alarm was raised. Because we gradually occupied system memory, every 100MB per second, it is reasonable that the negative impact of memory overuse faults cannot observed immediately. For this experiment, the detection latency was $\Delta = 7 \times 4 - 12 = 16$ seconds, and the true positive rate was $A_{td} = \frac{3}{3+4} \approx 0.43$.



(a) External I/O request rate (b) $corr(req, mem)$

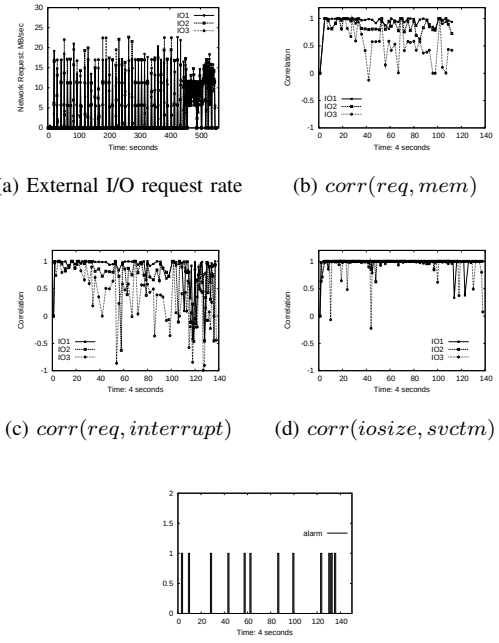
(c) $corr(req, interrupt)$ (d) $corr(iosize, svctm)$

Figure 14: Memory overuse faults were injected at 12nd second, the workload was 1GB sequential write on PVFS.

E. Benchmark Workload

The efficiency and accuracy of the detector have been demonstrated with synthetic workloads in different faulty situations; however, it is still necessary to examine the behaviors of the detector with realistic workloads to comprehensively evaluate it. In this section, a parallel I/O benchmark, BTIO, was adopted to evaluate the detector. BTIO is a tool contained in the NAS Parallel Benchmarks (NPB), and it is used to test the output capabilities of high-performance computing systems, especially distributed storage systems [27]. In our experiments, BTIO was complied with four processes, these processes work cooperatively to perform I/O operations on a dedicated storage system.

Figures 15 shows the results of BTIO test on PVFS in fault-free situation. For the test on PVFS, the invariants for memory and disk held at most time through the experiment; however, the invariant for CPU of all I/O servers did not hold well, especially, the invariant was frequently broken after the 115 sampling period in Figure 15c, meanwhile, a significant drop of req was also observed at the same time in Figure 15a. It is because BTIO performed read operations which requires frequently synchronization among all processes after the sampling period that more CPU time was occupied for synchronization. It is necessary to point out that there was no value generated by I_{mem} after the 116th sampling period, as there was few memory allocations after that. Because no fault was injected for the test, any alarm generated by the detector was marked as false detection. In the Figure 15e, there were a total of 12 alarms raised through the test, thus, the true positive rate was $A_{td} = \frac{126}{126+12} = 0.91$.



(a) External I/O request rate (b) $corr(req, mem)$

(c) $corr(req, interrupt)$ (d) $corr(iosize, svctm)$

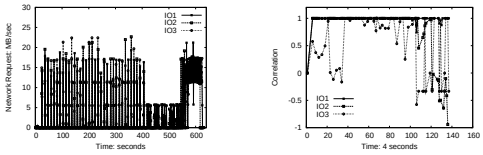
(e) Alarms

Figure 15: BTIO on PVFS.

1) Disk Delay Faults: This set of experiments evaluated the detector in the case of disk delay faults. The delay was set to 50 ms for the following experiments. Figures 16 shows the results of BTIO test on PVFS where the disk delay faults were introduced at IO2.

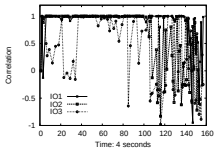
In Figure 16, disk delay faults were injected at the 410th second at IO2. The detector correctly detected the performance problem caused by the faults at the 109th sampling period. The detection latency was $\Delta = 109 \times 4 - 410 = 26$ seconds. The big latency was largely due to the discontinuous workloads generated by BTIO, as there was few incoming requests on IO3 between the 103rd and 109th sampling period. In Figure 16e, the detector raised four false alarms before the correct one, thus, the true positive rate was $A_{td} = \frac{111}{111+4} \approx 0.97$.

2) Network Delay Faults: This set of experiments evaluated the detector in the case of network delay faults. The delay was set to 50 ms for the following experiments. Figure 17 shows the results of BTIO test on PVFS where

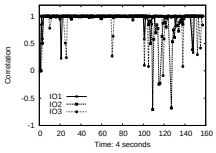


(a) External I/O request rate

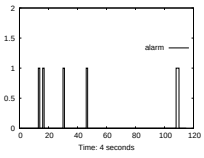
(b) $corr(req, mem)$



(c) $corr(req, interrupt)$

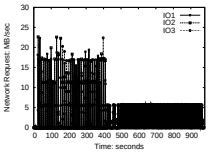


(d) $corr(iosize, svctm)$

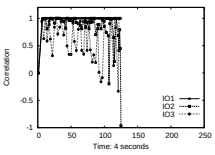


(e) Alarms

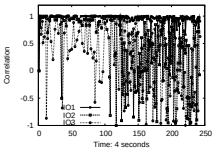
Figure 16: PVFS: Disk delay faults were injected at the 410th second at IO2.



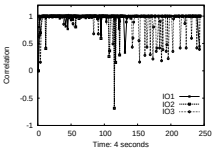
(a) External I/O request rate



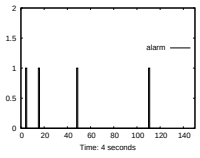
(b) $corr(req, mem)$



(c) $corr(req, interrupt)$



(d) $corr(iosize, svctm)$



(e) Alarms

Figure 17: PVFS: Network delay faults were injected at the 411th second at IO2.

the disk delay faults were introduced at IO2.

In Figure 17, network delay faults were injected at the 411th second at IO2. The detector correctly detected the performance problem caused by the faults at the 110th sampling period. The detection latency was $\Delta = 110 \times 4 - 411 = 29$ seconds. The big latency was largely due to the discontinuous workloads generated by BTIO, as there was no incoming requests on IO3 between the 100th and 108th sampling period. Meanwhile, the invariant for CPU of all I/O servers were more frequently broken after the 100th sampling period than the normal case, which can be regarded as a side effect of disk delay faults. In Figure 17e, the detector raised three false alarms before the correct one, thus, the true positive rate was $A_{td} = \frac{117}{117+3} \approx 0.98$.

F. Summary

Table V gives a summary of experiments with synthetic and BTIO workloads. For synthetic workloads, the detection latency is limited to two sampling periods (8 seconds), the average true positive rate is 84%, and there are no more than two false detections for most tests except the ones of memory overuse. The main reason for the poor performance of our detector in the experiments of memory overuse is that we gradually occupied system memory, our detector was insensitive to the small memory leak, as system performance was not significantly affected until a large portion of memory resource was leaked, thus our detector cannot detect immediately the faults of memory overuse.

Because the workloads generated by BTIO were not continuous, the average detection latency in the experiments with BTIO is larger than the one in the experiments with synthetic workloads. However, the average accuracy of 94% can be achieved by the detector for BTIO workloads.

VI. Conclusion

In this work, we presented a performance anomaly detector which is used to detect performance anomaly and accurately identify the faulty sources in an I/O server of cluster file systems. We concluded three invariants of an I/O server, which referred to the stable relationships between server workloads and resource statistics when the server works properly. By utilizing these invariants, a performance detector was developed, and the detector was evaluated with synthetic and BTIO workloads on PVFS file system in the presence of four different faulty situations. Our preliminary results demonstrated the efficiency and accuracy of the detector.

Acknowledgment

A preliminary version of this work was presented at the 3rd International Symposium on Parallel Architectures, Algorithms, and Programming (PAAP) [28].

This research is sponsored in part by National Science Foundation grants CNS-1102629, CCF-1102605, and CCF-1102624. This work is also partially supported by the SeaSky Scholar fund of the Dalian University of Technology. Any opinions, findings, and conclusions or recommendations expressed in this material are those of

TABLE V.: A Summary of Experiments with Synthetic Workloads.

Filesystem	Workload	Fault	Detection Latency	True positive rate	# of false detection
PVFS	1GB write	Disk delay	3 seconds	100%	0
		CPU overuse	5 seconds	83%	1
		Memory overuse	17 seconds	43%	4
		Network delay	5 seconds	80%	1
	1GB read	Disk delay	7 seconds	67%	2
		CPU overuse	7 seconds	80%	1
		Memory overuse	17 seconds	43%	4
		Network delay	6 seconds	80%	1
	BTIO	Disk delay	26 sec	97%	4
		Network delay	29 sec	98%	3

the author(s) and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] L. Cherkasova, K. M. Ozonat, N. Mi, J. Symons, and E. Smirni, "Anomaly? Application Change? or Workload Change? Towards Automated Detection of Application Performance Anomaly and Change," in *DSN '08: Proceedings of the International Conference on Dependable Systems and Networks*, June 2008, pp. 452–461.

[2] M. P. Kasick, J. Tan, R. Gandhi, and P. Narasimhan, "Black-Box Problem Diagnosis in Parallel File Systems," in *FAST '10: Proceedings of the 8th conference on File and Storage Technologies*, February 2010, pp. 57–70.

[3] X. Chen, J. Langston, and X. He, "An Adaptive I/O Load Distribution Scheme for Distributed Systems," in *PMEO-UCNS' 10: The 9th International Workshop on Performance Modeling, Evaluation, and Optimization of Ubiquitous Computing and Networked Systems in conjunction with IPDPS'10*, April 2010.

[4] R. D. Schlichting and F. B. Schneider, "Fail-stop processors: an approach to designing fault-tolerant computing systems," *ACM Trans. Comput. Syst.*, vol. 1, no. 3, pp. 222–238, 1983.

[5] E. S. Buneci and D. A. Reed, "Analysis of Application Heartbeats: Learning Structural and Temporal Features in Time Series data for Identification of Performance Problems," in *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, November 2008, pp. 1–12.

[6] H.-L. Truong, P. Brunner, T. Fahringer, F. Nerieri, R. Samborski, B. Balis, M. Bubak, and K. Rozkwitalski, "K-WfGrid Distributed Monitoring and Performance Analysis Services for Workflows in the Grid," in *E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*, 2006, pp. 1–15.

[7] H. Chen, G. Jiang, and K. Yoshihira, "Failure Detection in Large-Scale Internet Services by Principal Subspace Mapping," *IEEE Trans. on Knowl. and Data Eng.*, vol. 19, no. 10, pp. 1308–1320, 2007.

[8] Z. Lan, Z. Zheng, and Y. Li, "Toward Automated Anomaly Identification in Large-Scale Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, pp. 174–187, 2009.

[9] S. Kavulya, R. Gandhi, and P. Narasimhan, "Gumshoe: Diagnosing Performance Problems in Replicated File-Systems," in *SRDS '08: Proceedings of the 2008 Symposium on Reliable Distributed Systems*, October 2008, pp. 137–146.

[10] Z. Guo, G. Jiang, H. Chen, and K. Yoshihira, "Tracking Probabilistic Correlation of Monitoring Data for Fault Detection in Complex Systems," in *DSN '06: Proceedings of the International Conference on Dependable Systems and Networks*, June 2006, pp. 259–268.

[11] J. Gao, G. Jiang, H. Chen, and J. Han, "Modeling Probabilistic Measurement Correlations for Problem Determination in Large-Scale Distributed Systems," in *ICDCS '09: Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems*, June 2009, pp. 623–630.

[12] B. Chun and A. Vahdat, "Workload and Failure Characterization on a Large-Scale Federated Testbed," *Intel Research Berkeley Technical Report IRB-TR-03-040*, November 2003.

[13] D. Oppenheimer, A. Ganapathi, and D. A. Patterson, "Why do internet services fail, and what can be done about it?" in *USITS'03: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems*, 2003, p. 1.

[14] R. K. Sahoo, A. Sivasubramaniam, M. S. Squillante, and Y. Zhang, "Failure Data Analysis of a Large-Scale Heterogeneous Server Environment," in *DSN '04: Proceedings of the 2004 International Conference on Dependable Systems and Networks*, June 2004.

[15] P. Yalagandula, S. Nath, H. Yu, P. B. Gibbons, and S. Seshan, "Beyond Availability: Towards a Deeper Understanding of Machine Failure Characteristics in Large Distributed Systems," in *First Workshop on Real Large Distributed Systems (WORLDS)*, December 2004.

[16] B. Schroeder and G. A. Gibson, "A Large-Scale Study of Failures in High-Performance Computing Systems," in *DSN '06: Proceedings of the International Conference on Dependable Systems and Networks*, June 2006, pp. 249–258.

[17] W. Jiang, C. Hu, Y. Zhou, and A. Kanevsky, "Are disks the Dominant Contributor for Storage Failures? a Comprehensive Study of Storage Subsystem Failure Characteristics," in *FAST'08: Proceedings of the 6th USENIX Conference on File and Storage Technologies*, February 2008, pp. 1–15.

[18] L. N. Bairavasundaram, G. R. Goodson, B. Schroeder, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dussea, "An Analysis of Data Corruption in the Storage Stack," in *FAST'08: Proceedings of the 6th USENIX Conference on File and Storage Technologies*, February 2008, pp. 233–238.

[19] W. Jiang, C. Hu, S. Pasupathy, A. Kanevsky, Z. Li, and Y. Zhou, "Understanding Customer Problem Troubleshooting from Storage System Logs," in *FAST '09: Proceedings of the 7th conference on File and storage technologies*, February 2009, pp. 43–56.

[20] G. Jiang, H. Chen, and K. Yoshihira, "Discovering Likely Invariants of Distributed Transaction Systems for Autonomic System Management," in *ICAC '06: Proceedings of the 2006 IEEE International Conference on Autonomic Computing*, 2006, pp. 199–208.

[21] M. Placek and R. Buyya, "A Taxonomy of Distributed Storage Systems," www.cloudbus.org/reports/DistributedStorageTaxonomy.pdf.

[22] X. Chen, J. Langston, and X. He, "Design and Evaluation of a User-Oriented Availability Benchmark for Distributed File Systems," in *PDCS '09: Proceedings of the 21st IASTED International Conference on Parallel and Distributed Computing and Systems*, November 2009.
[23] "PVFS," March 2009, http://www.pvfs.org/.
[24] "Lustre," July 2008, http://en.wikipedia.org/wiki/Lustre_(file_system).
[25] D. P. Bovet and M. Cesati, *"Understanding the Linux Kernel: From I/O Ports to Process Management, 3nd"*. O'Reilly Media, Inc., 2006, iSBN: 0-596-00565-2.
[26] Y. Hu, A. Nanda, and Q. Yang, "Measurement, Analysis and Performance Improvement of the Apache Web Server," in *Proceedings of the IEEE International Performance, Computing, and Communications Conference*, 1999, pp. 261–267.
[27] P. Wong and R. F. V. der Wijngaart, "NAS Parallel Benchmarks I/O Version 2.4," 2003, http://www.nas.nasa.gov/News/Techreports/2003/PDF/nas-03-002.pdf.
[28] X. Chen, X. He, H. Guo, and Y. Wang, "An Online Performance Anomaly Detector in Cluster File Systems," in *PAAP '10: Proceedings of the 3rd International Symposium on Parallel Architectures, Algorithms, and Programming*, December 2010.

Xin Chen received both the MS and PhD degrees in electrical engineering from Tennessee Technological University, USA, in 2007 and 2010, respectively, and his BS degree in Mechanical Engineering from Shanghai University, China in 2003. He is currently a system engineer at Dell Inc., Austin, Texas, USA. His research interests include computer architecture and storage systems.

Xubin He received the PhD degree in electrical engineering from University of Rhode Island, USA, in 2002 and both the BS and MS degrees in computer science from Huazhong University of Science and Technology, China, in 1995 and 1997, respectively. He is currently an associate professor in the Department of Electrical and Computer Engineering at Virginia Commonwealth University. His research interests include computer architecture, storage systems, virtualization, and high availability computing. He received the Ralph E. Powe Junior Faculty Enhancement Award in 2004 and the TTU Chapter Sigma Xi Research Award in 2005 and 2010. He is a senior member of the IEEE and a member of the IEEE Computer Society.

He Guo received both the BS and MS degrees in computer science from Jilin University, China, in 1982 and Dalian University of Technology, China, in 1989, respectively. He is currently a professor in the Software School at Dalian University of Technology. His research interests include computer architecture, virtualization, parallel computing and computer vision. He is a member of the IEEE Computer Society.

Yuxin Wang received both the BS and MS degrees in computer science from Dalian University of Technology, China, in 1997 and 2000, respectively. He is currently a lecturer in the School of Computer Science and Technology at Dalian University of Technology. His research interests include computer architecture, virtualization and pattern recognition.

Multi-pattern Matching with Wildcards

Meng Zhang*, Yi Zhang†, Jijun Tang‡ and Xiaolong Bai*
*College of Computer Science and Technology, Jilin University, Changchun, China
Email: zhangmeng@jlu.edu.cn, baixiaol@sina.com
†Department of Computer Science, Jilin Business and Technology College, Changchun, China
Email: whdzy2000@vip.sina.com
‡Department of Computer Science & Engineering, Univ. of South Carolina, USA
Email: jtang@cse.sc.edu

***Abstract*—Multi-pattern matching with wildcards is to find all the occurrences of a set of patterns with wildcards in a text. This problem arises in various fields, such as computational biology and network security. But the problem is not extensively studied as the single pattern case and there is no efficient algorithm for this problem. In this paper, we present efficient algorithms based on the fast Fourier transform. Let $P = \{p^1, \ldots, p^k\}$ be a set of patterns with wildcards where the total length of patterns is $|P|$, and a text t of length n over alphabet $a_1, \ldots, a_\sigma$. We present three algorithms for this problem where patterns are matched simultaneously. The first algorithm finds the matches of a small set of patterns in the text in $O(n \log |P| + occ \log k)$ time where occ is the total number of occurrences of P in t. The words used in the algorithm are of size $k\lceil 2 \lg \sigma \rceil + \sum_{i=1}^{k} \lceil \lg |p^i| \rceil$ bits. The second algorithm is based on a prime number encoding. It runs in time $O(n \log m + occ \log k)$ where m is the length of the longest pattern in P. The algorithm uses words with $k\lceil \lg(2m\sigma^2 + k^2) \rceil$ bits. The third one finds the occurrences of patterns in the text in time $O(n \log |P| \log \sigma + occ \log k)$ by computing the Hamming distance between patterns and the text. The algorithm uses words with $\sum_{i=1}^{k} \lceil \lg |p^i| \rceil$ bits. Moreover, we demonstrate an FFT implementation based on the modular arithmetic for machines with 64-bit word. Finally, we show that these algorithms can be easily parallelized, and the parallelized algorithms are given as well.**

***Keywords*-Algorithm; Multi-pattern matching; Wildcards; FFT.**

I. INTRODUCTION

The problem of multi-pattern matching with wildcards is to find a set of patterns $P = \{p^1, \ldots, p^k\}$ in a text t (both the text and the patterns allow to contain wildcards). Throughout the paper, k denotes the number of patterns, n denotes the length of t, Σ denotes the alphabet of σ symbols from which the symbols in P and t are chosen. The single pattern matching with wildcards problem has received much attention. Fischer and Paterson [12] presented the first solution based on the fast Fourier transforms (FFT). The running time is $O(n \log m \log \sigma)$ where m is the length of the pattern. Indyk [13] latter introduced a randomized $O(n \log n)$ time Monte Carlo algorithm. Kalai [14] gave a simpler and faster $O(n \log m)$ time algorithm. In 2002 the first deterministic $O(n \log m)$ time solution was presented by Cole and Hariharan [7]. It uses one convolution, and each symbol in the text and the pattern is encoded with a pair of rational numbers. Clifford and Clifford [5] recently gave a simpler deterministic algorithm with the same time complexity that uses three convolutions where the numbers used are as large as $4m(\sigma-1)^4/27$. By allowing to preprocess the text, Rahman and Iliopoulos [20] gave efficient solutions without using FFTs and developed an algorithm runing in time $O(n + m + occ)$ where occ is the total number of occurrences of P in t. Very recently, Linhart and Shamir [17] presented the prime number encoding. By this approach, if $m^\sigma = n$, the algorithm runs in $O(n \log m)$ time by computing a single convolution.

Much research focus on Multi-string matching problem. The first algorithm to solve this problem in $O(n \log \sigma)$ time is presented by Aho and Corasick [1] which generalizes the Knuth-Morris-Pratt algorithm [15]. The Commentz-Walter [8] algorithm is a direct extension of the Boyer-Moore algorithm [4] which also combined the idea of AC algorithm. Several parallel multi-string matching algorithms are presented that are either precise [10] or approximate [27], [25], [24], [26]. The factor recognition approach based algorithms [9], [19], [2] use either suffix automata or factor oracles for precise or weak factor recognition. For short patterns, bit parallelism leads to algorithms that are efficient in practice, see [18].

However, the problem of matching a set of patterns with wildcards is not extensively studied as the single pattern case. To date, there is no efficient algorithm for this problem. But multi-pattern matching problem arises in many applications, such as intrusion detection systems [29], anti-virus systems [28] and computational biology [17]. A close but different problem: matching a set of patterns with variable length don't cares was solved by Kucherov and Rusinowitch [16]. They proposed an algorithm that runs in time $O((n + L(P))\log L(P))$, where $L(P)$ is the total length of keywords in every pattern of the pattern set P, and $|t|$ is the length of the input text. A faster solution was given by [22] which runs in $O((n + \|P\|)\log \kappa / \log\log \kappa)$ time, where $\|P\|$ is the total number of keywords in all the patterns in P, and κ is the number of distinct keywords in all the patterns in P.

In this paper, we focus on the problem of matching a set of patterns with wildcards without preprocessing the text. We present three FFT based algorithms for

A preliminary version of this paper was presented at PAAP 2010 [23].

doi:10.4304/jsw.6.12.2391-2398

this problem. The first one extends the Clifford and Clifford algorithm [5] to handle multi-pattern and runs in $O(n \log |P| + occ \log k)$ time where $|P|$ denotes the total length of all the patterns in P. It can find the matches of a small set of patterns in the text by three convolutions. The words used in the algorithm are of size $k\lceil 2 \lg \sigma \rceil + \sum_{i=1}^{k} \lceil \lg |p^i| \rceil$ bits. The second one uses the prime number encoding to encode both the pattern and the text. It runs in time $O(n \log m + occ \log k)$ where m is the length of the longest pattern. The algorithm uses words with $k\lceil 2 \lg \sigma + \lg m + \lg \lg(m\sigma^2) \rceil$ bits. The drawback of the two methods is that when σ, $|P|$ and k are large, the word will be too long to fit into a machine word of modern processors (normally 32 or 64 bits). To shorten the word length, we present an algorithm that uses words with $\sum_{i=1}^{k} \lceil \lg |p^i| \rceil$ bits. The algorithm finds the occurrences of patterns in the text in time $O(n \log |P| \log \sigma + occ \log k)$ by computing the Hamming distances between the patterns and the text. The distances are computed by $2\lceil \lg \sigma \rceil$ convolutions. Moreover, we discuss the modular arithmetic based FFT and give all the necessary parameters for the FFT on the 64-bit architecture. The algorithms presented in this paper can be easily parallelized. On a q-processor PRAM model, the time complexity of the algorithms decreases by q times compared with that on a single processor.

The paper is organized as follows. Section II gives some basic notions. Section III presents the algorithms for multi-pattern matching with wildcards using Euclidean distance. In Section IV, we give the approach based on Hamming distance of bit vectors. Section V introduces the FFT based on modular arithmetic on 64-bit architectures. Some interesting issues are discussed in Section VI.

II. Preliminaries

Let Σ be a finite alphabet and $'*'$ the wildcard symbol. Denote by $|s|$ the length of a string s. A text $t = t[1] \dots t[n]$ and a pattern $p = p[1] \dots p[m]$ are strings over $\Sigma \cup \{'*'\}$. Given a pattern p and a text t, which both may contain wildcards, we say that p occurs at location j in t if:

$$p[i] = t[i+j-1] \text{ or } p[i] =' *' \text{ or } t[i+j-1] =' *', \quad \text{for } 1 \le i \le m. \quad (1)$$

We use $P = \{p^1, \dots, p^k\}$ to denote a set of patterns with wildcards. We use " $\cdot$ " to denote the concatenation of two patterns, for example $p^1 \cdot p^2$ is the concatenation of pattern p^1 and p^2. For an integer array $x = x[1], x[2], \dots, x[n]$, we use $x[i_1..i_2]$ where $1 \le i_1 \le i_2 \le n$ to denote the array $x[i_1], \dots, x[i_2]$ and use $2^e x$ to denote the integer array of length n, such that

$$(2^e x)[i] = x[i]2^e, \text{ for each } 1 \le i \le n. \quad (2)$$

The following definition is a basic technique used in this paper.

Convolution: The convolution, or cross-correlation, of two vectors a, b is the vector $a \oplus b$ such that $(a \oplus b)[i] = \sum_{j=1}^{|a|} a[j]b[(i+j-1) \bmod |b|], 1 \le i \le |b|$. Note that this definition of convolution involves wrap-around (i.e., b is assumed to be a cyclic vector).

Our algorithms are based on FFTs. An important property of FFT is that in the RAM model, $p \oplus t$ can be computed in $O(n \log n)$ time. By a standard trick [12], the running time can be further reduced to $O(n \log m)$. First, split the text into n/m pieces of length $2m$. The starting positions of the pieces are in the set $\{lm + 1 \mid 0 \le l < n/m\}$. The convolution between the pattern and each piece of the text is computed using FFT in time $O(m \log m)$ per piece. The overall time complexity is $O((n/m)m \log m) = O(n \log m)$.

III. Euclidean distance based multi-pattern matching with wildcards

In this section, we extend the wildcard matching algorithm of Clifford and Clifford [5] to multi-pattern. Generally speaking, the Clifford and Clifford algorithm first encodes each symbol by a unique positive number and replaces wildcards by 0's. Then, for each location $1 \le i \le n - m + 1$ in the text, the algorithm computes

$$\sum_{j=1}^{m} p[j]t[i+j-1](p[j] - t[i+j-1])^2 = \sum_{j=1}^{m} (p[j]^3 t[i+j-1] - 2p[j]^2 t[i+j-1]^2 + p[j]t[i+j-1]^3) \quad (3)$$

in $O(n \log m)$ time using FFTs. Wherever there is an exact match this sum will be exactly 0.

The numbers computed by the algorithm are as large as $4m(\sigma-1)^4/27$. In [6], the authors modified the algorithm as follows. First by replacing non-wildcards by 1's and wildcards by 0's in the text and the pattern, we get p' and t' respectively. Then, for each location $1 \le i \le n-m+1$ in the text, the algorithm computes

$$\sum_{j=1}^{m} p'[j]t'[i+j-1]\big(p[j] - t[i+j-1]\big)^2. \quad (4)$$

The result can be viewed as the squire of the Euclidean distance between the pattern and the substring starting from a location i in t. The maximal numbers used in the convolutions are reduced to $m\sigma^2$.

A. Algorithm 1

In our approach, for a pattern p and each location $1 \le i \le n$ in the text, we use the following wrap-around sum

$$d(p,t)[i] = \sum_{j=1}^{|p|} p'[j]t'[l(i,j)]\big(p[j] - t[l(i,j)]\big)^2 \quad (5)$$

where $l(i,j)$ denotes $(i+j-1) \bmod n$. We can compute (5) by the following formula which uses only three

convolutions.

$$\sum_{j=1}^{|p|} p'[j]t[l(i,j)]^2 - 2\sum_{j=1}^{|p|} p[j]t[l(i,j)] + \sum_{j=1}^{|p|} t'[l(i,j)]p[j]^2. \quad (6)$$

Wherever there is an exact match this sum will be exactly 0.

To match a set of patterns $p^1, \ldots, p^k$, we first construct a composed pattern of length $|P|$:

$$p = p^1 \cdot p^2 \cdot \ldots \cdot p^k.$$

Define $\alpha_1 = 0$, $\alpha_j = \sum_{l=1}^{j-1}(\lceil \lg |p^l| + 2\lg\sigma \rceil)$, for $1 \le j \le k$ and $o_1 = 0$, $o_j = \sum_{l=1}^{j-1} |p^l|$, for $1 \le j \le k$. We use $I[1..l]$ to denote the array of length n where all the entries are 1's. We construct I^P as follows

$$I^P = (2^{\alpha_1} I[1..|p^1|]) \cdot (2^{\alpha_2} I[1..|p^2|]) \cdot \ldots \cdot (2^{\alpha_k} I[1..|p^k|]).$$

Then we compute the following

$$R[i] = \sum_{j=1}^{|p|} I^P[j]p'[j]t'[l(i,j)]\big(p[j] - t[l(i,j)]\big)^2$$

$$= \sum_{j=1}^{|p|} I^P[j]p'[j]t[l(i,j)]^2 - 2\sum_{j=1}^{|p|} I^P[j]p[j]t[l(i,j)] + \sum_{j=1}^{|p|} I^P[j]p[j]^2 t'[l(i,j)]. \quad (7)$$

R can be computed using three convolutions. By checking whether the bit vector from $(\alpha_j + 1)$th significant bit to α_{j+1}th significant bit of the binary code of $R[i]$, denoted by $R[i]_{[\alpha_j+1..\alpha_{j+1}]}$, is all 0's, we will know whether p^j occurs at position $(i + o_j)$ mod n in t. That is to say, assume that t is a cyclic vector, then for each $|P|$-length factor of t starting from each position of t, the result of the matching of each pattern is stored in a disjoint bit interval in a word. We give the algorithm in Figure 1.

According to (7), we can see that for the resulting array R computed by Algorithm 1,

$$R[i] = \sum_{j=1}^{k} d(p^j, t)[i + o_j]2^{\alpha_j}, \text{ for } 1 \le i \le n. \quad (8)$$

For any $p^j \in P$, according to (5), we have $d(p^j, t)[i + o_j] \le |p^j|\sigma^2$. So $\alpha_{j+1} - \alpha_j$ (that equals $\lceil \lg |p^j| + 2\lg\sigma \rceil$) bits are enough to represent $d(p^j, t)[i + o_j]$. As a result, the binary code of $d(p^j, t)[i + o_j]$ equals $R[i]_{[\alpha_j+1..\alpha_{j+1}]}$. Thus we can get $d(p^j, t)[i + o_j]$ by computing $(R[i] \bmod 2^{\alpha_{j+1}})/2^{\alpha_j}$.

To verify the correctness of Algorithm 1, suppose that a pattern $p^j \in P$ occurs in the text t starting from position x. We have $d(p^j, t)[x] = 0$. If p^j does not occur in t starting from x, we have $d(p^j, t)[x] \neq 0$. Let $i = x - o_j$. Thus $R[i]_{[\alpha_j+1..\alpha_{j+1}]} = 0$ indicates $d(p^j, t)[i + o_j] = 0$, that is, p^j occurs at position $(i + o_j)$ mod n of t.

The algorithm takes $O(n \log |P|)$ time to compute R and uses $O(nk)$ time to check R to find whether there is any occurrence of patterns. Each entry of R uses $k\lceil 2\lg\sigma \rceil + \sum_{i=1}^{k} \lceil \lg |p^i| \rceil$ bits. The size of the words used in the FFTs are of the same size. If σ, $|P|$ and k are small enough, each word can fit into a single machine word of modern processors that is typically 32 bits or 64 bits. For example, for DNA sequences where $\sigma = 4$, Algorithm 1 can process four patterns each with 16 symbols or three patterns each with 64 symbols. But when σ, $|P|$ and k are not small the words used in FFTs have to be very long. For example, let $\sigma = 256$, even for two patterns, Algorithm 1 uses words exceeding 32 bits. The algorithm in Section IV tries to shorten the word size to cope with texts on larger alphabets and pattern sets that have larger number of patterns and longer length.

Algorithm 1

Input: Text t and pattern set $P = \{p^1, p^2, \ldots, p^k\}$.

1: $R \leftarrow \{0, 0, \ldots, 0\}$
2: $\alpha_1 \leftarrow 0, o_1 \leftarrow 0$
3: **for** $j \leftarrow 2$ to k **do**
4: $\quad \alpha_j \leftarrow \alpha_{j-1} + \lceil \lg |p^{j-1}| + 2\lg\sigma \rceil$
5: $\quad o_j \leftarrow o_{j-1} + |p^{j-1}|$
6: **end for**
7: $L \leftarrow o_k + |p^k|$
8: $p \leftarrow p^1 \cdot p^2 \cdot \ldots \cdot p^k$,
9: $I^P[i] = (2^{\alpha_1} I[1..|p^1|]) \cdot (2^{\alpha_2} I[1..|p^2|]) \cdot \ldots \cdot (2^{\alpha_k} I[1..|p^k|])$
10: Compute $(p^i)'$ where $1 \le i \le k$ and t' by replacing non-wildcards by 1's and wildcards by 0's in the t and p^i.
11: For $1 \le i \le n$, compute $R[i] = \sum_{j=1}^{L} I^P[j]p'[j]t[(i + j - 1) \bmod n]^2 - 2\sum_{j=1}^{L} I^P[j]p[j]t[(i + j - 1) \bmod n] + \sum_{j=1}^{L} I^P[j]t'[(i + j - 1) \bmod n]p[j]^2$ using FFT.
12: **for** $pos \leftarrow 1$ to n **do**
13: $\quad$ **for** $j \leftarrow 1$ to k **do**
14: $\quad\quad$ **Output** "p^j occurs at $(pos + o_j)$ mod n in t" if $R[pos]_{[\alpha_j+1..\alpha_{j+1}]} = 0$.
15: $\quad$ **end for**
16: **end for**

Fig. 1. Algorithm 1.

The time complexity of checking the matches of P in t can be further reduced. Let the length of the longest pattern in P be m. Checking the matches of P in t can be done in time $O(n \log(m\sigma) + occ \log k)$ where occ is the times of occurrences of patterns in t. We first transform R to array ϱ such that $\varrho[i]_{[l]} = 0$ if $l \notin \{\alpha_1 + 1, \alpha_2 + 1, \ldots, \alpha_k + 1\}$ and for $1 \le j \le k$,

$$\varrho[i]_{[\alpha_j+1]} = \begin{cases} 1 & \text{if } R[i]_{[\alpha_j+1..\alpha_{j+1}]} = 0; \\ 0 & \text{if } R[i]_{[\alpha_j+1..\alpha_{j+1}]} \neq 0. \end{cases}$$

For p^j position $\alpha_j + 1$ is called the indication position (id position) of p^j. The transformation is described as

follows. Let there be d different lengths of patterns. We order the lengths in an increasing order, use m_j where $1 \leq j \leq d$ to denote the jth minimal length. First, set $\varrho[i] = 0$ for $1 \leq i \leq n$. Then for each pattern p^j we compute the bit $\bigwedge_{e=\alpha_j+1}^{\alpha_{j+1}} \overline{R[i]}_{[e]}$ and set this bit to $\varrho[i]_{[\alpha_j+1]}$. It follows that if $R[i]_{[\alpha_j+1..\alpha_{j+1}]} = 0$ then $\bigwedge_{e=\alpha_j+1}^{\alpha_{j+1}} \overline{R[i]}_{[e]} = \varrho[i]_{[\alpha_j+1]} = 1$. The computation is by bitwise `shiftright` and bitwise `and` operations. In the transformation, We use d bit masks, say $Vmask[1], \ldots, Vmask[d]$. The bit mask $Vmask[j]$ is a word where the bits on the indication locations for patterns whose length is m_j are set to bit 1 and other bits are set to 0s. The transformation is given in Figure 2. The time complexity of the algorithm is $O(n \log(m\sigma^2))$

Transform R
Input: An array R of length n.

1: **for** $i \leftarrow 1$ to n **do**
2: $R[i] \leftarrow \overline{R[i]}$
3: **end for**
4: **for** $j \leftarrow 1$ to d **do**
5: $Vmask[j] = 0$
6: **for** each pattern p^e such that $|p^e| = m_j$ **do**
7: $Vmask[j]_{[\alpha_e+1]} \leftarrow 1$
8: **end for**
9: **end for**
10: **for** $i \leftarrow 1$ to n **do**
11: $j \leftarrow 1$, $x \leftarrow Rr \leftarrow R[i]$, $RE \leftarrow 0$
12: **for** $s \leftarrow 1$ to $\lceil \lg(m\sigma^2) \rceil - 1$ **do**
13: $Rr \leftarrow Rr >> 1$
14: $x \leftarrow x \ \wedge \ Rr$
15: **if** $s = \lceil \lg(m_j\sigma^2) \rceil - 1$ **then**
16: $RE \leftarrow x \ \wedge \ Vmask[j]$
17: $j \leftarrow j + 1$
18: **end if**
19: **end for**
20: $\varrho[i] \leftarrow RE$
21: **end for**

Fig. 2. Transform R to ϱ.

When ϱ is available, we next check each entry of ϱ to find matches. For an entry $x = \varrho[i]$, we use an implicit binary tree T to find the id positions on which the bit values 1. Each tree node corresponds to a subset of the indication positions. A node $u = [n_1..n_2]$ consists of the indication positions of p_j where $j \in [n_1..n_2]$. The root is the set of all id positions, denoted by $[1..k]$. Each leaf contains one id position. For node u, the two children are $[n_1..(n_1+n_2)/2]$ and $((n_1+n_2)/2..n_2]$. We compute a bit mask $Mask(u)$ from u as follows: the bits of $Mask(u)$ on positions in $[n_1..n_2]$ are set to 1s, other bits are set to 0s.

We start at the root of T and check $x \wedge Mask([1..k])$. If it is 0, then there is no match for patterns $p^1, p^2, \ldots, p^k$. If it is not 0, at least one pattern matches, we continue to search in the left subtree of the root by checking $x \wedge Mask([1..k/2])$. If it is 0 then there is no match for patterns $p^1, p^2, \ldots, p^{k/2}$, and we prune the left branch; otherwise, at least one pattern matches, so we continue to search the left subtree of the root. After searching in the left subtree, we search in the right subtree. In this manner we traverse T depth-first, checking $x \wedge Mask(u)$ for each visited node u and pruning the branch if it is 0 along the way. At last, all the occurrences will be found by visiting the leaves corresponding to the matched patterns.

The time complexity is $O(n + occ \log k)$. So the total time complexity for finding the matches in R is $O(n \log(m\sigma^2) + occ \log k)$. For $|P| > m$ and $|P| \geq \sigma$, the total time complexity of Algorithm 1 is $O(n \log |P| + occ \log k)$.

B. Prime number encoding based algorithm

In this section, we introduce another strategy for matching a set of patterns with wildcards. Other than concatenating the patterns to a long one, this method aligns the patterns and generates a composed pattern with the length of the longest pattern. The algorithm is based on a prime number encoding of patterns. Let m be the length of the longest pattern in P. We extend each pattern to a similar length m by padding $'*'$s to the end of a pattern. Denote the resulting pattern set by P'. By padding m 0's to the end of the input, any matching of P in t is exactly a matching of P' in t for the same pattern on the same position.

We first pick up k distinct prime numbers $\rho_1, \rho_2, \ldots, \rho_k$. Denote $M = \rho_1 \cdot \rho_2 \cdots \rho_k$. We further require that ρ_i are larger than $|p^i|\sigma^2$.

Now we consider the encoding method. For k non-negative integers $x_1, x_2, \ldots, x_k$, where each x_i is not greater than σ, define an integer X that is less than M, such that for $1 \leq i \leq k$

$$X \equiv x_i \pmod{\rho_i}. \tag{9}$$

According to the Chinese Remainder Theorem (CRT, in short) [11], define $c_i = M/\rho_i\big((M/\rho_i)^{-1} \bmod \rho_i\big)$. For c_i, we have $c_i \equiv 1 (\bmod \rho_i)$ and $c_i \equiv 0 (\bmod \rho_{i'}), i' \neq i$. By the CRT, $X = \sum_{i=1}^{k} c_i \cdot x_i$.

We construct a composed pattern γ of length m from P, where

$$\gamma[i] = \sum_{j=1}^{k} c_j \cdot p^j[i], \text{ for } 1 \leq i \leq m, \tag{10}$$

and another composed pattern γ' of length m, where

$$\gamma'[i] = \sum_{j=1}^{k} c_j \cdot (p^j)'[i], \text{ for } 1 \leq i \leq m. \tag{11}$$

The text is encoded as follows:

$$\tau[i] = \sum_{j=1}^{k} c_j \cdot t[i], \text{ for } 1 \leq i \leq n. \tag{12}$$

τ' is encoded as follows:

$$\tau'[i] = \sum_{j=1}^{k} c_j \cdot t'[i], \text{ for } 1 \le i \le n. \quad (13)$$

Since $\rho_j > |p^j|\sigma^2$ and $d(p^j, t)[i] \le |p^j|\sigma^2$, for $1 \le j \le k$, we have

$$\gamma \bmod \rho_j = p^j, \tau \bmod \rho_j = t. \quad (14)$$

Then we can use Clifford and Clifford algorithm to match P' in t as follows. We compute Equation (5) for γ and τ and get array $d(\gamma, \tau)$. For an arbitrary array A, define $A \bmod p$ as the array of the same length of A where $(A \bmod p)[i] = A[i] \bmod p$. According to the CRT, for $1 \le j \le k$, we have

$$\begin{aligned} d(\gamma, \tau)[i] \bmod \rho_j &= d(\gamma \bmod \rho_j, \tau \bmod \rho_j)[i] \\ &= d(p^j, t)[i], \text{ for } 1 \le i \le n. \end{aligned} \quad (15)$$

Suppose that a pattern $p^j \in P$ occurs in the text t starting from position i. We have $d(p^j, t)[i] = 0$. So, $d(\gamma, \tau)[i] \bmod \rho_j = 0$. If p^j does not occur in t starting from x, we have $d(\gamma, \tau)[i] \bmod \rho_j \neq 0$. To find the matches of patterns, we have to check every entry of $d(\gamma, \tau)$, say $d(\gamma, \tau)[i]$, for all ρ_js such that $d(\gamma, \tau)[i] \bmod \rho_j = 0$. This straight forward method can find the occurrences in $O(nk)$ time. However, by using the method of Linhart and Shamir [17], the time complexity can be reduced to $O(n + occ \log k)$.

For $x = d(\gamma, \tau)[i]$, we use an implicit binary tree T to find ρ_j such that $x \bmod \rho_j = 0$. Each tree node corresponds to a subset of the pattern set. The root is the set of all patterns, denoted by $[1..k]$. Each leaf contains one pattern. For a node $u = [n_1..n_2]$, the two children are $[n_1..(n_1+n_2)/2]$ and $((n_1+n_2)/2..n_2]$. For node u, we compute an integer $Modul(u)$ as follows: $Modul(u) = \rho_{n_1} \cdot \rho_{n_1+1} \cdots \rho_{n_2}$.

We start at the root of T and check $x \bmod M$. If it is not 0, then there is no match for patterns. If it is 0, at least one pattern matches, we continue to search in the left subtree of the root by checking $x \bmod \rho_1 \cdot \rho_2 \cdots \rho_{k/2}$. If it is not 0 then there is no match for patterns $p^1, p^2, \ldots, p^{k/2}$, and we prune the left branch; otherwise, at least one of these patterns matches, so we continue to search the left subtree of the root. We traverse T in a depth-first manner, pruning some of the branches along the way. In the end, all the occurrences will be found by visiting the leaves corresponding to the matched patterns.

The algorithm takes $O(n \log m)$ time to compute R and uses $O(n + occ \log k)$ time to check R to find whether there is any occurrence of patterns. The total running time is $O(n \log m + occ \log k)$.

Now we consider the size of words used in the algorithm. For the number $\pi(x)$ of primes not exceeding x, the following bound is well known:

$$\forall x > 17 \quad \frac{x}{\ln x} < \pi(x) < 1.26\frac{x}{\ln x}. \quad (16)$$

According to the bound, we can verify that for $m\sigma^2 > 17$,

$$\pi(2m\sigma^2 + k^2) - \pi(m\sigma^2) > \frac{2m\sigma^2 + k^2}{\ln(2m\sigma^2 + k^2)} - 1.26\frac{m\sigma^2}{\ln(m\sigma^2)} > k. \quad (17)$$

Thus, if we search for prime numbers between $m\sigma^2+1$ and $2m\sigma^2 + k^2$, we will find at least k prime numbers. Using the sieve algorithm of Atkin et al. [3] it takes $o(2m\sigma^2 + k^2)$ time to find these prime numbers. Each prime number is at most $\lg(2m\sigma^2 + k^2)$ bits long. Therefore, $\lg M \le k \lg(2m\sigma^2 + k^2)$. Thus, each entry of R uses $k\lceil \lg(2m\sigma^2 + k^2) \rceil$ bits.

IV. Hamming distance based approach

A. Hamming distance of bit vectors with wildcards

Let $B = b_1 b_2 \ldots b_n$ be a binary pattern with wildcards. Denote $\overline{b_1 b_2} \ldots \overline{b}_n$ by $\overline{B}$ where if $b_i = *$ then $\overline{b}_i = *$. Given a binary pattern p and a bit string t (t is assumed to be a cyclic vector), such that both p and t have wildcards. The matchings between the pattern and a factor of t of length m have seven cases: 11, $1*$, $*1$, 00, $0*$, $*0$ and $**$ alignments. The Hamming distance between the two strings is the sum of the number of nonmatchings, that is $\#10 + \#01$.

The Hamming distance between p and the factor starting from each position of t of length m can be computed by the following method: For a bit vector v with wildcards, denote by $h^x(v)$ the bit vector where each wildcard of v is replaced by the bit x. Then $(h^0(\overline{p}) \oplus h^0(t))[i]$ equals $\#01$ between p and the factor of t starting from i, and $(h^0(\overline{t}) \oplus h^0(p))[i]$ equals $\#10$ between the same pair of strings. The Hamming distance between p and the factor of t starting from i can be computed by the following:

$$H(p, t)[i] = (h^0(\overline{p}) \oplus h^0(t))[i] + (h^0(\overline{t}) \oplus h^0(p))[i]. \quad (18)$$

This distance can be computed by two convolutions.

B. Multi-pattern matching by Hamming distance of bit vectors

We present an algorithm for multi-pattern matching with wildcards based on Hamming distance. In this approach, we derive $\lceil \lg \sigma \rceil$ bit vectors from the input text and $\lceil \lg \sigma \rceil$ patterns from the pattern and then perform $\lceil \lg \sigma \rceil$ times of pattern matchings for $\lceil \lg \sigma \rceil$ pairs of pattern and bit text. For $1 \le s \le \lceil \lg \sigma \rceil$, denote the sth bit of the integer encoding of a character $c \in \Sigma$ by $c_{[s]}$, for $c = *$, $c_{[s]} = *$. For a string t over Σ, denote the bit vector $t[1]_{[s]} t[2]_{[s]} \ldots t[n]_{[s]}$ by $t_{[s]}$.

To match a set of patterns, we first construct a series of patterns $B_{\langle 1 \rangle}$, $B_{\langle 2 \rangle}$, ..., $B_{\langle \lceil \lg \sigma \rceil \rangle}$ from P, such that for $1 \le s \le \lceil \lg \sigma \rceil$,

$$B_{\langle s \rangle} = 2^{a_1} p^1_{[s]} \cdot 2^{a_2} p^2_{[s]} \cdot \ldots \cdot 2^{a_k} p^k_{[s]},$$

where $a_1 = 0$, $a_j = \sum_{l=1}^{j-1} \lceil \lg |p^l| \rceil$.

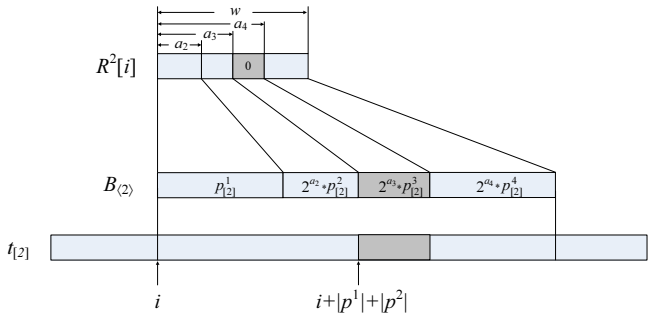


Fig. 3. An example of the run of Algorithm 3. Pattern p_2^3 is matched in t^2. The matching position is $i+|p^1|+|p^2|$ and $R^2[i][a_3+1..a_4]=0$.

Define

$$B_{\langle s,0\rangle} = 2^{a_1}h^0(p^1_{[s]}) \cdot 2^{a_2}h^0(p^2_{[s]}) \cdot \ldots \cdot 2^{a_k}h^0(p^k_{[s]}),$$
$$\widetilde{B}_{\langle s,0\rangle} = 2^{a_1}h^0(\overline{p^1_{[s]}}) \cdot 2^{a_2}h^0(\overline{p^2_{[s]}}) \cdot \ldots \cdot 2^{a_k}h^0(\overline{p^k_{[s]}}). \quad (19)$$

According to the computing of Hamming distance, for each $B_{\langle s\rangle}$ and P, we compute

$$R^s[i] = (\widetilde{B}_{\langle s,0\rangle} \oplus h^0(t_{[s]}))[i] + (h^0(\overline{t_{[s]}}) \oplus B_{\langle s,0\rangle})[i], \quad \text{for } 1 \le i \le n. \quad (20)$$

At last we compute the resulting array R such that

$$R[i] = R^1[i] \vee R^2[i] \vee \ldots \vee R^{\lceil \lg \sigma \rceil}[i], \text{ for } 1 \le i \le n. \quad (21)$$

By checking whether $R[i]_{[a_j+1..a_{j+1}]} = 0$, we will know whether p^j occurs at position $(i+o_j) \bmod n$ in t.

According to the definition of $B_{\langle s\rangle}$, we can see that for a resulting array R^s computed by the algorithm,

$$R^s[i] = \sum_{j=1}^{k} H(p^j_{[s]}, t_{[s]})[i+o_j]2^{a_j}, \text{ for } 1 \le i \le n. \quad (22)$$

For $p^j \in P$, we have $H(p^j_{[s]}, t_{[s]})[i+o_j] \le |p^j|$. So, $a_{j+1} - a_j = \lceil \lg |p^j| \rceil$ bits are enough to represent $H(p^j_{[s]}, t_{[s]})[i+o_j]$. Then we have

$$H(p^j_{[s]}, t_{[s]})[i+o_j] = R^s[i]_{[a_j+1..a_{j+1}]}. \quad (23)$$

Thus we can get $H(p^j_{[s]}, t_{[s]})[i+o_j]$ by computing $(R^s[i] \bmod 2^{a_{j+1}})/2^{a_j}$.

To see why this method works, suppose that a pattern $p^j \in P$ occurs in t starting from position x. We have $H(p^j_{[s]}, t_{[s]})[x] = 0$ for $1 \le s \le \lceil \lg \sigma \rceil$. Otherwise $H(p^j_{[s]}, t_{[s]})[x] \ne 0$ for at least one s. Let $x = i + o_j$. Since $R^s[i]_{[a_j+1..a_{j+1}]} = H(p^j_{[s]}, t_{[s]})[i+o_j]$, we have that $\bigvee_{s=1}^{\lceil \lg \sigma \rceil} R^s[i]_{[a_j+1..a_{j+1}]} = 0$ indicates $H(p^j_{[s]}, t_{[s]})[i+o_j] = 0$, for $1 \le s \le \lceil \lg \sigma \rceil$. Thus $R[i]_{[a_j+1..a_{j+1}]} = 0$ indicates that p^j occurs at position $(i+o_j) \bmod n$ of t. An example of the running of this algorithm is shown in Figure 3 where we only illustrate one bit vector.

The algorithm is given in Figure 4.

Algorithm 3 takes $O(n \log |P| \log \sigma)$ time to compute R and uses $O(nk)$ time to check whether there is any occurrence of patterns. By the method in Section III-A, the time for checking matches in R can be reduced to $O(n \log m + occ \log k)$. The total time complexity is therefore $O(n \log |P| \log \sigma + occ \log k)$. The words used in the algorithm are of size $w = \sum_{i=1}^{k} \lceil \lg |p^i| \rceil$ bits. If we use a $O(n \log m)$ time single pattern matching algorithm to match each pattern one by one, the total time is $O(n \sum_{i=1}^{k} \log |p^i| + occ \log k) = O(nw + occ \log k)$. Thus if $w > \lg |P| \lg \sigma$, Algorithm 3 takes less time for matching P against t than the time taken by single pattern matching algorithms. If $w \le \lg |P| \lg \sigma$, we can partition the pattern set into a set of subsets of P guaranteeing that $w > \lg |LP| \lg \sigma$ for each subset LP. We then match all these subsets of P using our algorithm one by one. The overall running time is certainly less than the time for running a single pattern algorithm for all the patterns.

If the occurrences of patterns are rare, we can revise Algorithm 3 to have a better performance. Recall that to use FFT efficiently, the input text is split into $n/|P|$ pieces of length $2|P|$ and each piece is matched against P independently. Then in the running of Algorithm 3, the pattern set is matched against every piece. Denote the piece starting at position $(l-1)|P|+1$ in t where

Algorithm 3
Input: text t and pattern set $P = \{p^1, p^2, \ldots, p^k\}$.

1: $R \leftarrow \{0, 0, \ldots, 0\}$
2: $a_1 \leftarrow 0, o_1 \leftarrow 0$
3: **for** $j \leftarrow 2$ to k **do**
4: $\quad a_j \leftarrow a_{j-1} + \lceil \lg |p^{j-1}| \rceil$
5: $\quad o_j \leftarrow o_{j-1} + |p^{j-1}|$
6: **end for**
7: $L \leftarrow o_k + |p^k|$
8: **for** $s \leftarrow 1$ to $\lceil \lg \sigma \rceil$ **do**
9: $\quad B_{\langle s,0\rangle} = 2^{a_1}h^0(p^1_{[s]}) \cdot 2^{a_2}h^0(p^2_{[s]}) \cdot \ldots \cdot 2^{a_k}h^0(p^k_{[s]})$
10: $\quad \widetilde{B}_{\langle s,0\rangle} = 2^{a_1}h^0(\overline{p^1_{[s]}}) \cdot 2^{a_2}h^0(\overline{p^2_{[s]}}) \cdot \ldots \cdot 2^{a_k}h^0(\overline{p^k_{[s]}})$
11: **end for**
12: **for** $s \leftarrow 1$ to $\lceil \lg \sigma \rceil$ **do**
13: $\quad$ For $1 \le i \le n$, compute $R^s[i] \leftarrow \sum_{r=1}^{L} \widetilde{B}_{\langle s,0\rangle}[r]h^0(t_{[s]})[(i+r-1) \bmod n] + \sum_{r=1}^{L} B_{\langle s,0\rangle}[r]h^0(\overline{t_{[s]}})[(i+r-1) \bmod n]$ using FFT.
14: $\quad$ **for** $i \leftarrow 1$ to n **do**
15: $\quad\quad R[i] \leftarrow R[i] \vee R^s[i]$
16: $\quad$ **end for**
17: **end for**
18: **for** $pos \leftarrow 1$ to n **do**
19: $\quad$ **for** $j \leftarrow 1$ to k **do**
20: $\quad\quad$ **Output** "p^j occurs at $(pos+o_j) \bmod n$ in t" if $R[pos]_{[a_j+1..a_{j+1}]} = 0$.
21: $\quad$ **end for**
22: **end for**

Fig. 4. Algorithm 3.

$1 \leq l \leq n/|P|$ by $piece(t, l)$. The resulting arrays computed by Algorithm 3 taking $piece(t, l)$ and P as input are denoted by $R^s(l)$, for each $1 \leq s \leq \lceil \lg \sigma \rceil$. We have $R^s(l)[i] = R^s[i + (l-1)|P|]$ where $1 \leq i \leq |P|$. Algorithm 3 is revised as follows. In computing R^s, for piece $piece(t_{[s]}, l)$, if $R^s(l)[i]_{[a_j+1..a_{j+1}]} \neq 0$ for all $1 \leq i \leq |P|$, then we have that no pattern occurs in piece $piece(t, l)$. In the followed computing, all the pieces $piece(t_{[s']}, l)$ where $s' > s$ are neglected. That is to say, $piece(t_{[s']}, l)$ will not be matched against $B_{\langle s' \rangle}$. The revised algorithm is correct for the situation that no pattern occurs in the neglected pieces. Since the occurrences of patterns are rare, the number of inspected pieces of bit vectors of the input by the revised algorithm is small. The cost is that in processing a bit vector of the input, the revised algorithm has to check the resulting array to determine which piece of the input can be neglected. In the original algorithm, the check is done only once when all the convolutions are finished.

V. FFT BASED ON INTEGER ARITHMETIC

In this section, we give an FFT implementation based on modular arithmetic other than complex numbers. In [21], Yap and Li present a fast integer multiplication algorithm based on an efficient FFT implementation built on the integer arithmetic. The approach aims at reducing the overhead in the implementation of FFT on machines in which the words are 32-bit or 64-bit. This FFT implementation fits the context of our multi-pattern matching algorithm well. The basic idea of the approach is to perform FFT in the ring $\mathbb{Z}_M = \{0, 1, \ldots, M-1\}$ of numbers modulo M. Here M is a specially chosen prime number. In a w-bit machine, M is at most w-bits so that the component-wise product can be done in $O(1)$ machine operations. For an n-length vector, in the field $\mathbb{Z}_M$, we need primitive n-th roots of unity. According to [21], the modular M is a prime number that can be expressed as

$$M = nd + 1. \tag{24}$$

We next pick up a primitive element of $\mathbb{Z}_M$, say e, and set

$$\omega = e^d \bmod M. \tag{25}$$

Then we have that each of $\omega^i \bmod M$ is distinct and $\neq 1$, for $i = 1, 2, 3, \ldots, n-1$. We have $w^n \equiv e^{nd} \equiv e^{M-1} \equiv 1(\bmod M)$ by Fermat's little theorem. That is ω is an n-th primitive root of unity.

In practice, to proceed the recursion of FFT, we need n to be a power of 2. For the case that the machine word is 32 bits, Yap and Li [21] choose $M_{32} = 2,013,265,921$ as the modulo, a prime number with only 31 bits that can be expressed as $M_{32} = 2^{27}d+1$ where $d = 15$ and $n = 2^{27}$. The number 31 can be proved to be a primitive element of $\mathbb{Z}_{M_{32}}$. The primitive 2^{27}-th roots of unity can be chosen as $\omega = 31^{15} \bmod M_{32} = 440,564,289$. Using ω, we can implement the FFT algorithm on a 32-bit machine, where we compute everything mod M_{32} and each component of the vectors is of the length 27 bits.

Based on Yap and Li's approach, we present the parameters for the 64-bit architecture. We choose $M_{64} = 6,269,010,681,299,730,433$ as the modulo, a prime number with only 63 bits. M_{64} has similar properties with M_{32}, which can be expressed as $M_{64} = 2^x d + 1$ where $d = 87$ and $x = 56$. Among the prime numbers that can be expressed as $2^x d + 1$, M_{64} has the maximal x. The number 5 turns out to be the smallest primitive element of $\mathbb{Z}_{M_{64}}$. The primitive 2^{56}-th roots of unity can be chosen as

$$\omega = 5^{87} \bmod M_{64} = 4,467,632,415,761,384,939.$$

Using ω, we can implement the FFT algorithm on a 64-bit machine, where we compute everything mod M_{64} and each component of the vectors is of the length 56 bits.

VI. PARALLELIZED ALGORITHMS

The algorithms in this paper can be easily parallelized. For Algorithm 1, we design a parallel multi-pattern matching algorithm with no communication. According to the trick in Section II, we first split the text into $n/|P|$ pieces each of length $2|P|$. The starting positions of the pieces are in the set $\{l|P| + 1 \mid 0 \leq l < n/|P|\}$. The convolution between the composed pattern and each piece of the text is computed using FFT in time $O(|P| \log |P|)$ per piece. We do not use the parallelized FFT, but use the sequential version of FFT conducted by processors. As each piece can be matched independently, no communication is needed. Since each piece is of length $2|P|$ and completely overlaps with the adjacent pieces, any occurrence of a pattern will keep integrate in at least one piece. Therefore, the parallelized Algorithm 1 is correct. On a q-processor PRAM model, the overall time complexity of parallelized Algorithm 1 is $O((n \log |P| + occ \log k)/q)$.

Algorithm 2 and Algorithm 3 can be parallelized in the same way as Algorithm 1. Readily, the time complexity of the parallelize Algorithm 2 and Algorithm 3 on a q-processor PRAM model are $O((n \log m + occ \log k)/q)$ and $O((n \log |P| \log \sigma + occ \log k)/q)$ accordingly.

VII. CONCLUSION AND FURTHER RESEARCH

We have presented three algorithms for multi-pattern matching with wildcards. The first one can find the matches of a small set of patterns in a text in $O(n \log |P| + occ \log k)$ time. The words used in the algorithm are of size $k\lceil 2 \lg \sigma \rceil + \sum_{i=1}^{k} \lceil \lg |p^i| \rceil$ bits. The second algorithm finds the occurrences of patterns in time $n \log m + occ \log k$ based on a prime number encoding of the pattern set and the text, using the words of $k\lceil \lg(2m\sigma^2 + k^2) \rceil$ bits. The last algorithm is based on Hamming distance between bit vectors. It runs in $O(n \log |P| \log \sigma + occ \log k)$ time and uses the words with $\sum_{i=1}^{k} \lceil \lg |p^i| \rceil$ bits. If the number of wildcards in the patterns is very small, the problem can be solved by building a deterministic finite automaton (DFA) that detects all possible words that match the pattern. One advantage of our algorithm over finite automata based algorithms is that the patterns in our algorithm need not

be preprocessed but taken as the input. In automata based approaches, the pattern set is used to build automata that will be further optimized for a low memory usage or a better performance. Whenever built, it is difficult to add or remove patterns from the existing data structures for the automata. In our approach, as patterns are taken as the input, adding or deleting a pattern has very low costs, thus our approach has more flexibilities.

It remains to determine whether there exists an $O(n \log |P|)$ algorithm using words of a proper size. That is in the resulting array, each bit of an entry indicates whether a pattern occurs at the corresponding location of the text. The modifications on FFT itself would be necessary to reach this goal.

ACKNOWLEDGMENTS

This work was in part supported by Fundamental Research Funds for the Central Universities No.200903186, Education Department of Jilin Province No.599, NSF of Science & Technology Department of Jilin Province No.20101522, NSF grant OCI 0904179, Chinese NSF No.60703024 and Program for New Century Excellent Talents in University NCET-09-0428.

REFERENCES

[1] A. V. Aho, M. J. Corasick: Efficient String Matching: An Aid to Bibliographic Search, *Comm. ACM* 18, 333–340, 1975.
[2] C. Allauzen, M. Raffinot. Factor oracle of a set of words. Technical Report 99-11, Institut Gaspard-Monge, Université de Marne-la-Vallée, 1999.
[3] A. Atkin, D. Bernstein. Prime sieves using binary quadratic forms, *Math. Comp.* 73, 1023–1030, 2004.
[4] R. S. Boyer, J. S. Moore. A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772, 1977.
[5] P. Clifford and R. Clifford, Simple deterministic wildcard matching. *Inf. Process. Lett.* 101(2): 53–54, 2007.
[6] R. Clifford, K. Efremenko, E. Porat and A. Rothschild, From coding theory to efficient pattern matching. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithm*, 778–784, 2009.
[7] R. Cole and R. Hariharan, Verifying candidate matches in sparse and wildcard matching. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 592–601, 2002.
[8] B. Commentz-Walter. A string matching algorithm fast on the average. In Proc. of the *6th International Colloquium on Automata, Languages and Programming*, LNCS 71, 118–132, 1979.
[9] M. Crochemore, A. Czumaj, L. Gąsieniec, T. Lecroq, W. Plandowski, and W. Rytter. Fast practical multi-pattern matching. *Information Processing Letters*, 71(3/4):107–113, 1999.
[10] Maxime Crochemore, Zvi Galil, Leszek Gasieniec, Kunsoo Park, Wojciech Rytter. Constant-Time Randomized Parallel String Matching. *SIAM J. Comput*. 26(4): 950–960, 1997.
[11] T. Cormen, C. Leiserson, R. Rivest, C. Stein, Introduction to Algorithms, 2nd Edition, MIT Press and McGraw-Hill, 2001.
[12] M. Fischer and M. Paterson, String matching and other products. In *Proceedings of the 7th SIAM-AMS Complexity of Computation*, 113–125, 1974.
[13] P. Indyk, Faster algorithms for string matching problems. Matching the convolution bound. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, 166–173, 1998.
[14] A. Kalai, Efficient pattern-matching with don't cares. In Proceedings of the 13th *Annual ACM-SIAM Symposium on Discrete Algorithms*, 655–656, Philadelphia, PA, USA, 2002.
[15] D. E. Knuth, J. H. Morris, and V. R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(1):323–350, 1977.
[16] G. Kucherov and M. Rusinowitch, Matching a Set of Strings with Variable Length don't Cares. *Theor. Comput. Sci.* 178(1-2): 129–154, 1997.
[17] C. Linhart and R. Shamir, Faster pattern matching with character classes using prime number encoding. *J. Comput. Syst. Sci.*75(3): 155–162, 2009.
[18] G. Navarro, M. Raffinot. Flexible Pattern Matching in Strings – Practical on-line search algorithms for texts and biological sequences. Cambridge University Press, 2002.
[19] M. Raffinot. On the multi backward dawg matching algorithm (MultiBDM). In R. Baeza-Yates, editor, Proceedings of the *4th South American Workshop on String Processing*, 149–165, 1997.
[20] M. Rahman and C. Iliopoulos, Pattern Matching Algorithms with Don't Cares. *SOFSEM* (2), 116–126 2007.
[21] C. Yap and C. Li. QuickMul: Practical FFT-based Integer Multiplication. Technical report, Department of Computer Science, Courant Institute, New York University, October 2000.
[22] M. Zhang, Y. Zhang and L. Hu, A faster algorithm for matching a set of patterns with variable length don't cares. *Inf. Process. Lett.* 110(6): 216–220, 2010.
[23] M. Zhang, Y. Zhang and J. Tang, Matching a set of patterns with wildcards. *Third International Symposium on Parallel Architectures, Algorithms and Programming* (PAAP'10), 169–174, IEEE Computer Society, 2010.
[24] C. Zhong, Z. Fan and D. Su, Parallel Approximate Multi-Pattern Matching on Heterogenerous Cluster Systems, Proceedings of the *9th International Conference on Parallel and Distributed Computing, Appplications and Technologies*, 74–79, IEEE Computer Society Press, 2008.
[25] C. Zhong, D. Fan, Parallel Algorithms for Approximate String Matching with Multi-Round Distribution Strategy on Heterogeneous Cluster Computing Systems(in Chinese). *Journal of Computer Research and Development*,45(S1):105–112,2008.
[26] Z. Fan, C. Zhong, X. Cui, L. Xu, Parallel Algorithm for Approximate Multiple Object Strings Matching on Heterogeneous Cluster Computing Systems with Limited Memory(in Chinese), *Journal of Chinese Computer Systems*, 30(2):225–229,2009.
[27] C. Zhong, G. Chen, Parallel Algorithms for Approximate String Matching on PRAM and LARPBS(in Chinese), *Journal of Software*, 15(2):159–169, 2004.
[28] Clam AntiVirus. URL: `http://www.clamav.net`.
[29] Snort Intrusion Detection System. URL: `http://www.snort.org`.

BIOGRAPHIES

Meng Zhang received his Ph.D. in Computer Science from Jilin University, in 2003. He is currently an Associate Professor in College of Computer Science and Technology, Jilin University. His main research interests include stringology, network security and computational biology.

Yi Zhang received her Ph.D. in Computer Science from Jilin University, in 2009. She is currently an Associate Professor in Department of Computer Science, Jilin Business and Technology College. She is also a Postdoc researcher in Jilin university. Her main research interests include artificial intelligent and computational biology.

Jijun Tang Prof. Jijun Tang received his Ph.D. in Computer Science from University of New Mexico, in 2004. He is currently an Associate Professor in Department of Computer Science and Engineering, University of South Carolina. His main research interests include high performance algorithm development, computational biology and engineering simulation. During the past five years, His research has been supported by ONR, NSF and NIH.

Xiaolong Bai is currently a third year undergraduate student in College of Computer Science and Technology of Jilin University. His research interests include algorithm design and network security.

RPPA: A Remote Parallel Program Performance Analysis Tool

Yunlong Xu, Zeng Zhao, Weiguo Wu*, and Yixin Zhao
School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an, China
Email: xjtu.ylxu@gmail.com, wgwu@mail.xjtu.edu.cn

***Abstract*—Parallel program performance analysis plays an important role in exploring parallelism and improving efficiency of parallel programs. To ease the performance analysis for remote programs, this paper presents a remote parallel program performance analysis tool, RPPA (Remote Parallel Performance Analyzer), which is based on dynamic code instrumentation. A hierarchical structure is adopted by RPPA which consists of 3 parts: client, server and computing nodes. Performance analysis tasks are submitted to the server via the graphical user interface of the client, and then actual analysis processes are started on the computing nodes by the server to collect performance data for visualization in the client. The performance information gained by RPPA is comprehensive and intuitive, hence it is quite helpful for users to analyze performance, locate bottlenecks of programs, and optimize programs.**

***Index Terms*—parallel programming tools, program performance analysis, dynamic code instrumentation, performance visualization**

I. INTRODUCTION

In the past few years, parallel computers have grown very fast, while software for parallel computing has grown comparatively slow. This situation makes the efficiency of parallel systems relatively low, thus hardware performance cannot be fully utilized [1]. Therefore software becomes the bottleneck of parallel computing, and limits the widely use of parallel computers. For instance, parallel program development is challenging due to the lack of effective tools for coding, debugging, performance analyzing and optimizing.

Motivated by the preceding challenge, a remote visualization parallel program performance analysis tool, RPPA, which aims to help programmers to optimize performance of applications and make full use of parallel computing resources, is designed and developed based on the survey of existing tools and our former research of a parallel program integrated development environment (i.e., IDE). RPPA provides a solution for performance analysis of MPI [4] applications running on parallel computers with SMP nodes. To gain comprehensive performance information of programs, a dynamic instrumentation based approach, which includes modifying, deleting, and inserting code to change the execution behaviors of programs, is applied to the performance measurement. The framework of RPPA consists of three parts: client, server and computing nodes. A performance analysis task is committed to the server through the client graphical user interface (i.e., GUI); a server daemon running on the server then starts the program performance data collection processes on several computing nodes; later, the server gathers performance data files from computing nodes, and sends back to the client for visualization.

The reminder of this article is organized as follows: First, we consider related work and point out RPPA's strength through comparison with existing tools in the next section. In Section 3, we describe the overall architecture of RPPA. After that, we present the methodology applied in the client, the server and the computing nodes in detail respectively in Section 4. In Section 5, we present the evaluation of a RPPA prototype. Section 6 is a discussion. Finally, we consider future work and conclude the paper.

II. RELATED WORK

Performance analysis is crucial to parallel program development. This work combines knowledge of various fields, e.g., parallel computing, computer architecture, data mining. The related work of program performance analysis has been carried out by many organizations and agencies.

MPE [2] is an extension of MPICH [3] which is a portable implementation of MPI. It consists of a large number of programming interfaces and examples for correctness debugging, performance analysis and visualization. It supports several file formats to log performance information of programs, which can be viewed by Upshot, Jumpshot [5], and other visualization tools.

Paradyn [6], developed by University of Wisconsin, is a software package which aids in analyzing performance of large-scale parallel applications. It supports analyzing MPI and PVM [7] applications. The cause of performance problems is systematically detected by inserting and modifying code of programs automatically. And performance bottlenecks are automatically searched by means of a W^3 (i.e., When, Where, and Why) model.

TAU [8] is a parallel program performance analysis system developed by University of Oregon, Juelich

The journal article is based on the conference paper, "Research and Design of a Remote Visualization Parallel Program Performance Analysis Tool," which appeared in PAAP'10.

*Corresponding author: Weiguo Wu

doi:10.4304/jsw.6.12.2399-2406

Research Center, and Los Alamos National Laboratory together. It focuses on system robustness, flexibility, portability and integration of other related tools, and supports multiple programming languages, including C/C++, Fortran, Java, Python. A MPI wrapper library is provided to analyze performance of MPI functions, as well as a log format conversion tool.

VENUS (Visual Environment of Neocomputer Utility System) [17] is a parallel program performance visualization environment developed by Xi'an Jiaotong University. It provides solutions for monitoring, analyzing and optimizing large-scale PVM parallel programs based on profiling method. System supports real-time and post-mortem performance visualization. Measurement code is automatically inserted into source code of programs to collect performance data for data analysis and performance visualization. In this way, performance analysis helps users to optimize their programs.

Open|SpeedShop [9], a dynamic binary instrumentation based performance tool using DPCL [16]/Dyninst [12], aims to overcome a common limitation of performance analysis tools: each tool alone is often limited in scope and comes with widely varying interfaces and workflow constraints, requiring different changes in the often complex build and execution infrastructure of the target application; thus it provides efficient, easy to apply, and integrated performance analysis for parallel systems.

HPCToolkit [10], developed by Rice University, is an integrated suite of tools that supports measurement, analysis, attribution, and presentation of application performance for both sequential and parallel programs. HPCToolkit can pinpoint and quantify scalability bottlenecks in fully-optimized parallel programs and multithreaded programs at a low cost. Call path profiles for fully-optimized codes can also be collected without compiler support.

The tools enumerated above have a common defect that performance visualization and data collection are both carried out on parallel computers or clusters. It is hard for users who are not familiar with parallel systems to install and apply these tools. The preceding defect hinders the widely use of parallel program development environment and parallel computing resources to some extent. Therefore, there is an urgent need for a user-friendly remote parallel program performance analysis tool. RPPA is to meet this need. Users connect to remote parallel systems over the internet via RPPA's client GUI, which is part of a parallel program IDE built on Eclipse Plug-in [11] technology. Interactive operations and performance visualization are both carried out in the client, while actual operations are executed on remote clusters which are transparent to common users. RPPA is easy to use and portable for Windows, Linux and other operating systems.

III. OVERALL ARCHITECTURE

The RPPA tool adopts a hierarchical structure, through which users are shielded from the complexity of parallel systems. The performance information of programs is presented in RPPA's GUI in an intuitive way, which is quite helpful to locate performance bottlenecks and improve efficiency of programs. A performance analysis task is committed to the server through the graphical client; a daemon running on the server starts the program performance data collection processes, whose core step is dynamic instrumentation, on several computing nodes; the server gathers performance data files from the computing nodes, and sends back to the client for visualization.

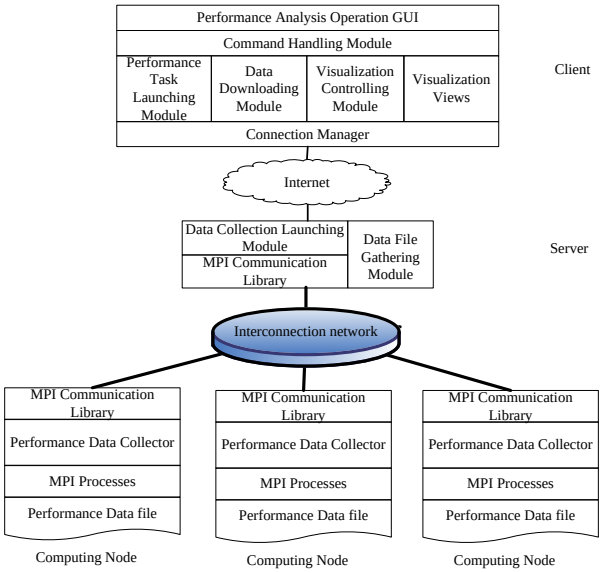


Figure 1. Overall architecture of RPPA

RPPA is customized to the cluster structure. The overall architecture of RPPA is depicted in Figure 1. The client connects with the server over the internet, and the computing nodes connect with the server, which can also be a computing node, over high-speed interconnection network. Users control the whole system through a GUI, which includes two core modules for submitting performance analysis tasks, downloading and visualizing performance data. The server-side operation details (e.g., how the run-time processes are assigned to the computing nodes) are transparent to users. As a result, operations are greatly simplified for users who are not familiar with the bottom parallel cluster structure. The server, entry node of a parallel cluster, is responsible for maintaining the connection between the client and the cluster, accepting instructions from the client and then starting performance analysis processes on computing nodes to collect performance data. Actual performance analysis tasks and program execution tasks are carried out on the computing nodes: RPPA creates user program processes, into which measurement code is inserted, controls its execution, generates performance data files, integrates and analyzes the files when analysis processes finish.

IV. DETAILED DESIGN

A. Client

Client provides an interface for users to commit

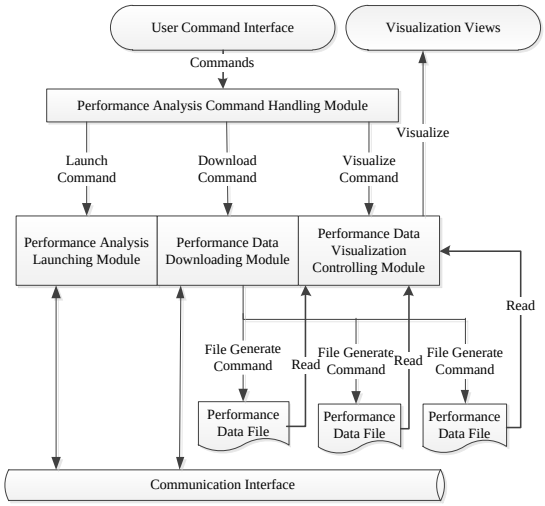


Figure 2. Structure of client

analysis tasks and visualizes performance analysis result. The GUI implemented with Eclipse Plug-in technology greatly simplifies users' operation. Client is composed of four functional modules: performance analysis command handling module, performance analysis launching module, performance data downloading module, and performance data visualization controlling module. The relationship between the four modules is shown in Figure 2.

Performance Analysis Command Handling Module

This module implemented by extending Plug-in extension points of Eclipse platform. Graphical items of performance analysis (e.g., buttons, menus, pop-up lists and editors) are added to the original GUI of Eclipse platform. Each item corresponds to a specific performance analysis operation. This module supports interactions between users and RPPA.

Performance Analysis Launching Module

This module sends commands to the server and starts remote performance analysis task. A customized Eclipse launching mode is implemented by extending the *launchModes* extension point of the original platform, and the actions carried out while launching are implemented by extending the *launchDelegates* extension point of Eclipse.

Performance Data Downloading Module

Performance data files are distributed among the computing nodes when analysis task finishes due to the C/S architecture adopted by RPPA. This module gathers and integrates the distributed performance data files, and then transfers the integrated file to the client for visualization.

Performance Data Visualization Controlling Module

The structure of this module is shown in Figure 3. Performance information can be viewed in multiple aspects through various display commands and views RPPA provides. A command deals with a specific kind of data set; and a view displays a kind of performance information. It is worth mentioning that, in accordance with the characteristics of parallel programs, a view of

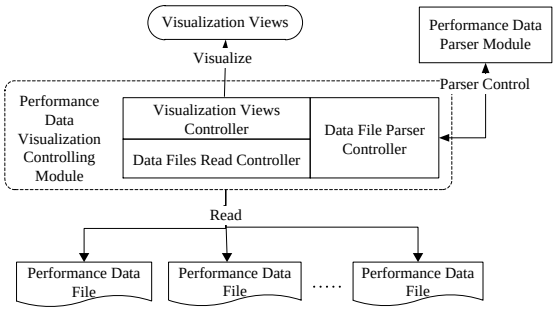


Figure 3. Performance visualization control module of client

performance comparison between multiple processes is provided.

Performance data visualization is an important means to help users analyze program performance and pinpoint performance bottlenecks. Because it is difficult to measure a program's performance based directly on large volume of performance data files of various entries, not to mention that the data volume grows rapidly as application scales. Therefore, RPPA offers vivid and intuitive graphical charts to help users visualize performance information analyze program performance and then locate bottlenecks quickly.

B. Server

Server is the communication relay between the client and the computing nodes: it serves as the entry of a cluster, which is composed of several computing nodes, and controls execution of tasks on computing nodes; it also receives commands from the client and responses to these commands. The functionalities of the server include launching performance analysis tasks, gathering performance data files from computing nodes, and maintaining communication with the client. The performance data file gathering module is the core of the server.

According to when collected data is analyzed and visualized, program performance analysis falls into two categories: real-time analysis and post-mortem analysis. Each mode has its strength: the real-time mode can adjust data collection during running time of tasks, while the post-mortem mode introduces minor perturbation into the original program. As to RPPA, if a real-time mode is adopted in the C/S structure, the server has to receive performance analysis instructions frequently, deal with these instructions, and then transmit them to the computing nodes on which they are actually executed, and finally large volume of data is transferred back to the client while execution. The whole process which is greatly affected by network would introduce major perturbation to the original program. Thus the post-mortem mode is adopted in RPPA.

Unlike the launch process of common MPI parallel program, the launch process of performance analysis task has to ensure that all MPI processes running on the computing nodes are under the control of the program performance analysis tool, specifically the control of the performance analysis launching module. In this way, measurement code is instrumented into user programs, and then performance data can be collected. The launch

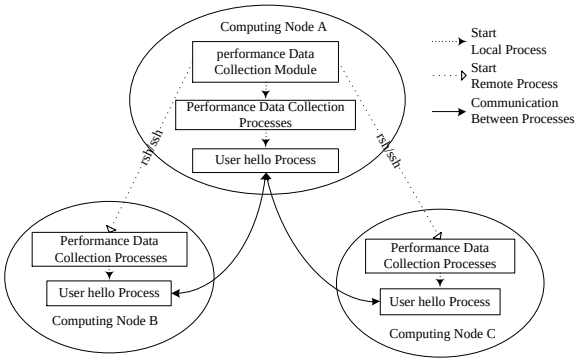


Figure 4. MPI program starting process under the control of data collection module

process of MPI parallel program under the control of launching module is shown in Figure 4. When receiving a performance analysis launch command, the launching module of the server (e.g., Node A in Figure 4) analyzes the parameters of the command, creates a performance analysis task, sets some parameters (e.g., name and full path of the user program, number of processes) for the task, and then starts data collection processes on the computing nodes. The data collection processes analyze the parameters and then create hello processes based on the name of the user processes and the parameters of the performance analysis task. After that, the hello processes is considered to be part of the whole task, and managed by MPI runtime environment.

C. Computing Nodes

The computing nodes, on which MPI runtime environment is deployed, is the substrate of RPPA. A bijection is set up between MPI processes and performance data collection processes on the computing nodes. Data collection processes start and control MPI processes. The performance data collection module of the computing nodes is launched by the performance data collection launching module of the server. MPI processes are started by the data collection module according to some parameters (e.g., the absolute path of the MPI program). And then the collection module inserts measurement code into MPI processes, and integrates generated performance data files when finishes.

The main purpose of performance analysis is to help users to locate program bottlenecks, i.e., code segments which account for large portions of running time. Besides running time, the number of function calls is also collected to estimate the average running time of each function. As to MPI parallel program, communication volume of MPI functions also concerns programmers a lot. To sum up, three kinds of data, i.e., running time, number of function calls, communication volume, need to be collected.

Performance data is collected by means of code instrumentation [12], which includes modifying, deleting, and inserting code to change the execution behaviors of programs. According to the timing of code inserting, instrumentation can be divided into two categories: static and dynamic code instrumentation. The static one inserts code before execution of programs and the dynamic one, per contra, during execution. The static instrumentation introduces little perturbation into the original programs, but the information it collects is not comprehensive; while the dynamic instrumentation is able to collect comprehensive performance information at the cost of major perturbation. Using dynamic instrumentation, programs are just awakened rather than to be recompiled, relinked and restarted after instrumentation. Apparently, the dynamic instrumentation is more complicated to implement than the static one.

In this paper, we use DynistAPI [12], a dynamic binary instrumentation interface developed by University of Maryland, to insert running processes with binary code. Instrumentation processes (i.e., Mutator) start user program processes which are to be instrumented (i.e., Mutatee), and attaches themselves to the user processes. Measurement code segments (i.e., Snippet) are inserted into user processes by instrumentation processes when user processes are suspended. And performance data is collected when instrumented code of user processes are executed.

The process of performance data collection is described as Algorithm 1. We note that, when no MPI

Algorithm 1: Performance data collection procedure

Input: *A*, Attributes of user task
Output: *PF*, an integrated performance data files

```
define: proc_array, array of MPI process pid
proc_array ← spawn(A)
suspend(proc_array)
foreach i in proc_array
    define: node, name of computing node
    define: file, path of data file
    file ← new(node,i)
    define: procImage, image of user process
    procImage ← getImage(i)
    define: usrModule, image of user process
    usrModule ← search(procImage)
    define: usrFunc_array, array of user function
    func_array ← search(usrModule)
    for j in func_array
        define: snippet, binary code segment
        define: gettimeofday, function to get current time
        define: fscanf/fprintf, function to read/write a file
        snippet ← generate(gettimeofday,fopen/fprintf)
        insert(j, snippet,file)
        if j is a MPI function
            define: commAttr_array, array of communication
                    attributes, e.g., volume, source, destination
            snippet ← generate(commAttr_array,file)
            insert(j,snippet)
        end if
    end for
    wake(proc_array)
    if i finishes
        acknowledge(server,file,i)
    end if
end foreach
while(TRUE)
    if all processes complete
        define: file_array, array of performance file
        PF ← integrate(file_array)
        return PF
    end if
end while
```

communication function is founded, the data collection process skips the step of collecting MPI communications information, and goes ahead rather than returns, because the program being analyzed could be a serial program, which can also be executed by *mpiexec* command in MPI runtime environments.

V. EVALUATION

This section describes the evaluation of a RPPA prototype in two aspects: functional evaluation and perturbation evaluation. Pros and cons of RPPA are also presented at the basis of the evaluations.

A. Test Environment and Test Case

In this paper, the evaluations are carried out on a parallel computing system, a Dawning 4000L cluster, which composed of 18 nodes. Each node is equipped with: Intel (R) Xeon (TM) CPU (3.00 GHz × 2), 2G memory, Broadcom BCM5721 1000Base-T LAN, Linux 2.4 operating system, and OpenMPI [14] 1.2.3 parallel computing environment. The test case is an MPI parallel program solving ocean circulation problem by using two-dimensional *stommel* model developed by Timothy H. Kaise.

B. Functional Evaluation of Performance Visualization

Users can check performance information in 7 different views, including: function running time of single process, function call numbers of single process, communication volume of single process, function running time comparison of multi-processes, function call number comparison of multi-processes, communication volume comparison of multi-processes, and space-time diagram of all processes. In this evaluation, the test program is composed of 5 performance data collection processes, each of which creates a *stommel* process on 2 computing nodes separately. Data files are downloaded and performance information is visualized when data collection task finishes. Function running time of one process is shown in Figure 5; function running time comparison of multi-processes is shown in Figure 6; and space-time diagram of all processes is shown in Figure 7.

Functional evaluation shows that the performance data collected by RPPA which is based on dynamic instrumentation is comprehensive, because performance data of all kinds of functions (i.e., user-defined functions, external library functions, standard MPI communication functions) can be collected through dynamic instrumentation. This evaluation also shows that, RPPA meets the requirement of program performance analysis, and brings great convenience to programmers by providing a user-friendly GUI.

C. Perturbation Evaluation

The perturbation introduced into the original programs by an analysis tool is an important metric to measure the performance of the analysis tool itself [15]. The perturbation can be reflected by the running time of programs. And the degree of perturbation can be estimated by comparing the running time of a program with measurement code inserted and the running time of the original program.

What concerns us most is the difference of running time between the original program and the program with code inserted. The running process of parallel programs is affected by many factors. And in this paper, perturbation evaluation is influenced mainly by network status and system background workload status. Thus a single node, on which there is no process running besides Linux system processes, is used to run the evaluation programs. In this way, the network and system load factors are ruled out.

The Paradyn performance analysis tool, which also uses the Dyninst tool for dynamic instrumentation, is selected to compare with RPPA. To make the evaluation data more representative, we carry out 3 kinds of tests using the *stommel* program, including running program without interference of any performance analysis tools, with interference of RPPA, and with interference of the Paradyn tool. 10 groups of these 3 kinds of tests are conducted. The test results are shown in Figure 8. Test type 1 represents running time without interference of any analysis tools, type 2 represents running time with RPPA, and type 3 represents the Paradyn tool.

It can be seen that, under the same conditions (i.e., the same running environment, and the same types of performance data to be collected), the running time of the stommel program with interference of RPPA is closer to the running time without any tools than to the running time with Paradyn. From this perspective, RPPA surpasses the Paradyn program performance analysis tool.

However, Figure 8 also shows that both RPPA and Paradyn introduce major perturbations into the original programs. The primary cause is that they both build on dynamic instrumentation which intrudes the original programs frequently by coping, modifying, transferring, and inserting code during runtime of programs. But for the same reason, the performance data collected by the analysis tools based on dynamic instrumentation is comprehensive, because performance information about all kinds of functions can be gained. Obviously, major perturbation is the price of comprehensive performance information.

VI. DISCUSSION

In this section, we discuss some new ideas and methods to improve RPPA. As mentioned in the previous section, dynamic instrumentation based performance analysis tool introduces major perturbation into the original programs. Hence we seek a novel approach to collect performance data on the server and computing nodes. A multi-level instrumentation based data collection approach is proposed in our recent research. The implementation of this approach is part of our recent and future work, and is planned to be presented in follow-up work. In this section, we first discuss why this approach would reduce the perturbation, and how it works.

As analysis in section 7 says, dynamic instrumentation collects comprehensive program performance

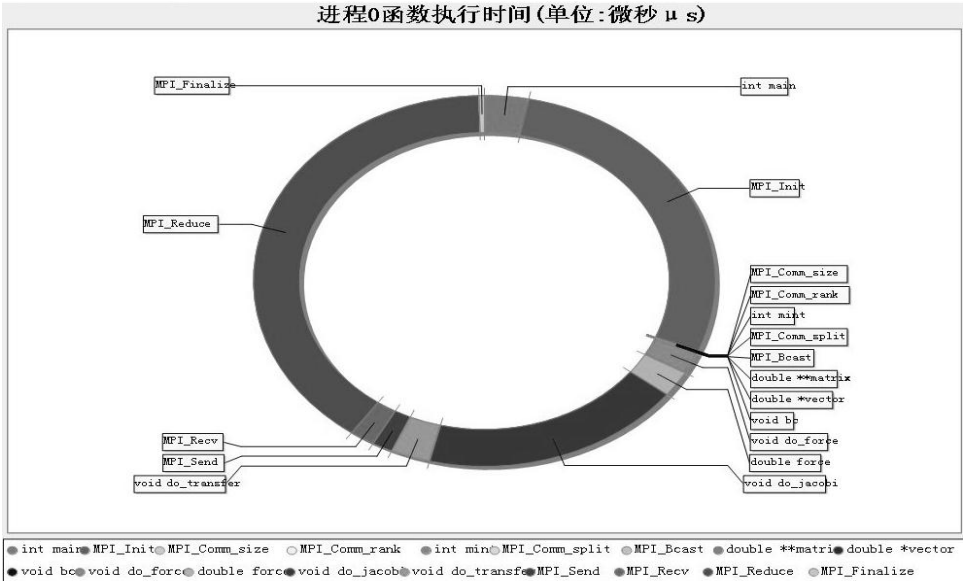


Figure 5. Function running time of one process

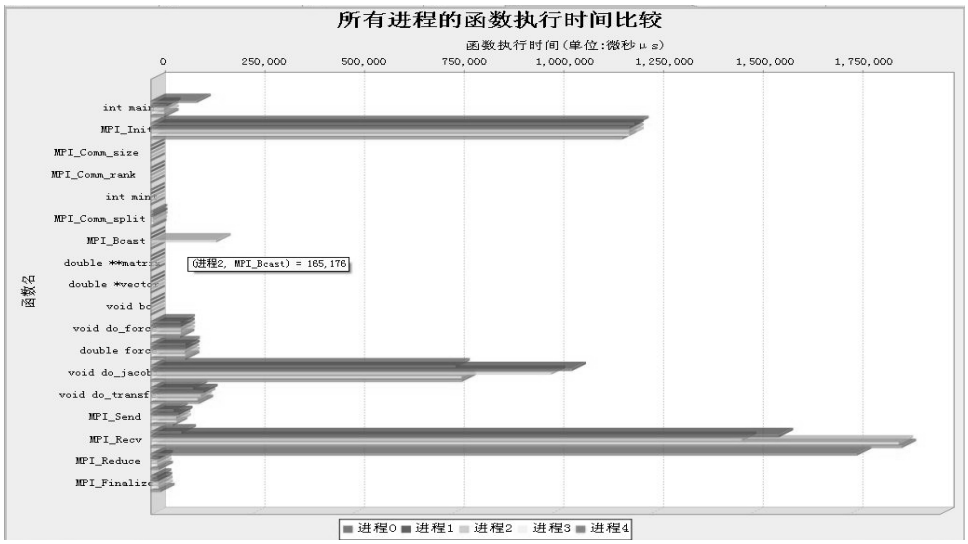


Figure 6. Function running time comparison of multi-processes

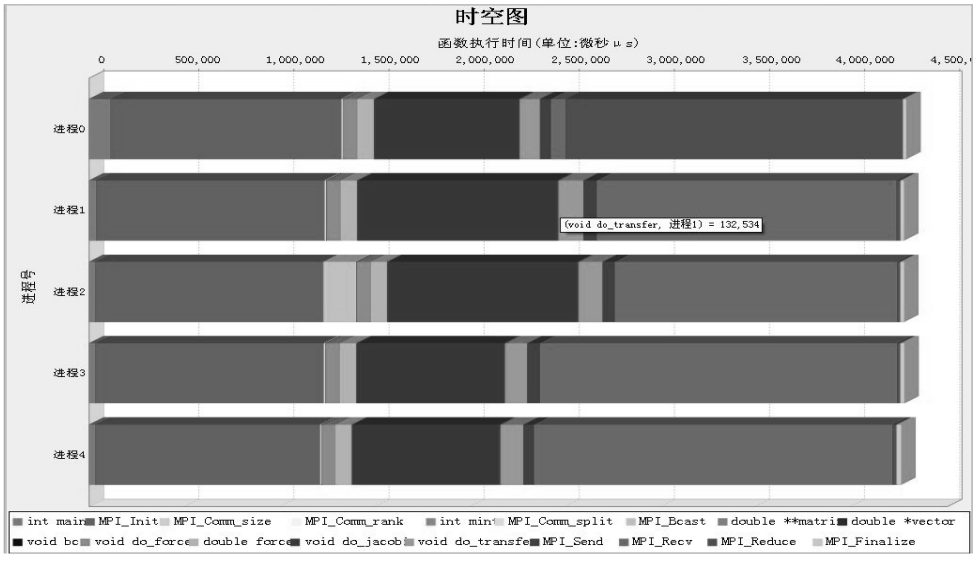


Figure 7. Space-time diagram of all processes

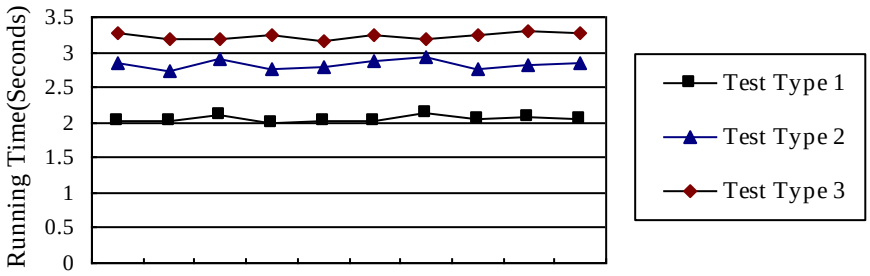


Figure 8. Running time comparison of *stommel* program interfered by different analysis tool

information at the cost of major perturbation. Therefore we propose the multi-level instrumentation based approach, through which the comprehensiveness of performance information would not be compromised. The essential of this approach is to collect different kinds of information in distinctive proper ways; and only do dynamic instrumentation when really necessary. We separate the functions, whose performance information concerns programmers most, into three categories: user-defined function, standard MPI function, external library function. Performance information of user-defined function is collected through a source-level instrumentation [8] based method. As to standard MPI functions, we adopt PMPI [4], the MPI standard profiling interface, to collect its performance information. And for external library function, to the best of our knowledge, the dynamic instrumentation is the best method to collect its performance information. We note that the source-level instrumentation could be replaced by the dynamic one when there are no other files than the binary ones for some reasons. We also note that dynamic instrumentation could be skipped if the external library functions' performance information is not a concern or little perturbation can be tolerated by users. And still there is plenty of performance information presented to programmers even if the external library functions are ignored.

By using this multi-level based approach, the perturbation introduced into the original programs would surely be at a lower level, and comprehensiveness of performance information would also not be compromised. Firstly, the source-level based instrumentation introduces little perturbation into the original programs because only a few measurement code is inserted into the source files at the entry and exit point separately when compilation. And all the information that the dynamic approach collects can also be collected by this source-level one. Secondly, when using PMPI profiling to collect standard MPI function's performance information, there is no need to copy, modify, transfer, and insert code during runtime of programs. Because all *MPI_* prefix functions are substituted by equivalent *PMPI_* ones in which some additional measurement code is added. Thus less perturbation is introduced. And also the comprehensiveness of performance information would not be compromised.

The philosophy of this tool resembles SaaS (i.e., Software-as-a-Service) in cloud computing which emerges and then receives significant attention in the media recently. The cloud conceals the complexity of the infrastructures from common users through internet. Cloud computing partly refers to the software delivered as services over the internet. Users could shift computing power away from local servers, across the network cloud, and into large clusters of machines hosted by companies such as Amazon, Google, IBM, Microsoft, Yahoo! and so on. We plan to transplant our parallel program IDE's client, which include RPPA's client, into the internet environment by rewriting its GUI with some web programming Languages. Then the Eclipse platform would be replaced by an internet explorer. And with the support of a daemon running on the server node of a remote cluster, users could develop parallel programs remotely over the internet on laptop rather than on parallel cluster. In this way, great convenience would be brought to programmers, while excellent portability of this tool would also be achieved.

VII. CONCLUSION AND FUTURE WORK

A remote parallel program performance analysis tool, RPPA, is designed based on the research of the existing program performance analysis tools, most of which are not user-friendly. RPPA meets the urgent need of parallel program performance analysis as mentioned in Section 1. The workflow of RPPA is as follows: performances analyses tasks are submitted to the server through the client; then performance data collection processes are started on the computing nodes by the server to collect performance data; at last, program performance information is visualized in the client. RPPA is user-friendly, easy to use, and the collected performance information is comprehensive, so it is quite helpful for users to analyze performance, locate bottlenecks of programs, and optimize programs.

However, there is a common shortcoming of dynamic instrumentation based performance analysis tools: major perturbation would be introduced into the original programs. So research on multi-level instrumentation based performance analysis tool which brings minor perturbation is a part of our future work. Our future work also includes providing supports for variety kinds of parallel programming models and heterogeneous high performance computing systems.

ACKNOWLEDGMENT

This work was supported by the National High-Tech Research and Development Plan of China under Grant No. 2009AA01A131 and No. 2009AA01A135; the Chinese Ministry of Science and Technology under Grant No. 2009DFA12110.

We would like to thank all the people who give helpful suggestions to our work, especially the reviewers of PAAP'10 for their comments on the original conference paper [18]. And we also would like to thank all the members of the integration and application of heterogeneous resources service project team, department of computer science and technology, Xi'an Jiaotong University, for their supports to our work.

REFERENCES

[1] C.C. Chen and M.X. Chen, "A generic parallel computing model for the distributed environment," *Proc. 2006 7th International Conference on Parallel and Distributed Computing, Applications and Technologies*, pp. 4-7, December 2006.

[2] A. Chan, W. Gropp, and E. Lusk, "User's guide for MPE: extensions for MPI programs," *Technical Report*, MCS-TM-ANL-98, Argonne National Laboratory, 1998.

[3] W. Gropp, "MPICH2: A new start for MPI implementations," *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, vol. 2474, pp. 37-42, 2002.

[4] Message Passing Interface Forum, "MPI: A message-passing interface standard," *Technical Report*, University of Tennessee, 1994.

[5] O. Zaki, E. Lusk, W. Gropp, and D. Swider, "Toward scalable performance visualization with Jumpshot," *International Journal of High Performance Computing Applications*, vol. 13, pp. 277-288, 1999.

[6] B.P. Miller, M. Callaghan, G. Cargille, J.K. Hollingsworth, R.B. Irvin, K.L. Karavanic, et al, "The Paradyn parallel performance measurement tool," *Computer*, vol. 28, pp. 37-46, November 1995.

[7] W. Gropp and E. Lusk, "Goals guiding design: PVM and MPI," *Proc. IEEE International Conference on Cluster Computing*, pp. 257-265, 2002.

[8] S. Shende and A.D. Malony, "The TAU parallel performance system," *International Journal of High Performance Computing Applications*, vol. 20, pp. 287-311, 2006.

[9] M. Schulz, J. Galarowicz, D. Maghrak, W. Hachfeld, D. Montoya and S. Cranford, "Open|SpeedShop: An open source infrastructure for parallel performance analysis," *Scientific Programming*, vol. 16, pp. 105-121, April 2008.

[10] L. Adhianto, S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, and N. R. Tallent, "HPCTOOLKIT: Tools for performance analysis of optimized parallel programs," *Concurrency and Computation: Practice & Experience*, vol. 22, pp. 1-7, April 2010.

[11] E. Clayberg and D. Rubel, "Eclipse Plug-Ins," *Addison-Wesley Professional*, 2008.

[12] B. Buck and J.K. Hollingsworth, "An API for runtime code patching," *International Journal of High Performance Computing Applications*, vol. 14, pp. 317-329, 2000.

[13] G. Ravipati, A. Bernat, N. Rosenblum, B.P. Miller, J.K. Hollingsworth, "Towards the deconstruction of Dyninst," *Technical Report*, University of Maryland, 2007.

[14] E. Gebrial, G. Fagg, G. Bosilca, T. Angskun, J. Dongarra, J. Squyres, et al, "Open MPI: goals, concept, and design of a next generation MPI implementation," *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, vol. 3241, pp. 97-104, 2004.

[15] J. Liu, M.M. Shen, and W.M. Zheng, "Research on perturbation imposed on parallel programs by debugger," *Chinese Journal of Computer*, vol. 25, No. 2, pp. 122-127, 2002.

[16] L. DeRose, T. Hoover and J. Hollingsworth, "The dynamic probe class library — an infrastructure for developing instrumentation for performance tools," *Proc. of the 15th International Parallel and Distributed Processing Symposium*, Apr. 2001.

[17] X.H. Shi, Y.L. Zhao, S.Q. Zheng, and D.P. Qian, "VENUS: A general parallel performance visualization environment," *Mini-micro Systems*, vol. 19, pp. 1-7, 1998.

[18] Y.L. Xu, Z. Zhao, W.G. Wu, and Y.X. Zhao, " Research and Design of a Remote Visualization Parallel Program Performance Analysis Tool," *Proc. of the Third International Symposium on Parallel Architectures, Algorithms and Programming*, pp. 214-220, Dec. 2010.

Yunlong Xu, born in Sichuan, China, 1987. He is currently a Ph.D. candidate at School of Electronic and Information Engineering, Xi'an Jiao tong University, China. His research interests include parallel computing, and cloud computing.

Zeng Zhao received his master degree in School of Electronic and Information Engineering, Xi'an Jiao tong University, China, in 2010. Zhao is currently a member of Tencent, Inc., China.

Weiguo Wu, born in 1963, is Professor, Ph.D. supervisor, of School of Electronic and Information Engineering, Xi'an Jiao tong University, China. He received his master and Ph.D. degree in Xi'an Jiaotong University. His research interests include high-performance computer architecture, massive storage system, computer networks, and embedded systems.

Yixin Zhao received his master degree in School of Electronic and Information Engineering, Xi'an Jiao tong University, China, in 2009.

Topic Detection with Hypergraph Partition Algorithm

Xinyue Liu[1, 2]
1. School of Computer Science and Technology, Dalian University of Technology, Dalian, China
2. School of Software, Dalian University of Technology, Dalian, China
xyliu@dlut.edu.cn

Fenglong Ma[2], Hongfei Lin[1]
1. School of Computer Science and Technology, Dalian University of Technology, Dalian, China
2. School of Software, Dalian University of Technology, Dalian, China
flma@mail.dlut.edu.cn, hflin@dlut.edu.cn

***Abstract*—An algorithm named SMHP (Similarity Matrix based Hypergraph Partition) algorithm is proposed, which aims at improving the efficiency of Topic Detection. In SMHP, a T-MI-TFIDF model is designed by introducing Mutual Information (MI) and enhancing the weight of terms in the title. Then Vector Space Model (VSM) is constructed according to terms' weight, and the dimension is reduced by combining H-TOPN and Principle Component Analysis (PCA). Then topics are grouped based on SMHP. Experiment results show the proposed methods are more suitable for clustering topics. SMHP with novel approaches can effectively solve the relationship of multiple stories problem and improve the accuracy of cluster results.**

***Index Terms*—topic detection, similarity matrix, hypergraph partition, clustering**

I. INTRODUCTION

Timely and efficiently access to amounts of information is becoming more and more significant. Gaining such access is no longer a problem owing to the widespread availability of Internet. However, it is difficult for users to locate interesting or useful information as the explosion in large volumes of digitized textual content. Evidently, it is impossible for humans to assimilate such vast quantities of information. Topic Detection has risen to challenge this difficulty.

Topic Detection, a subtask of Topic Detection and Tracing (TDT), is becoming a hot research interest in recent years. This subtask attempts to identify “topics” by analyzing and organizing the content of textual materials, thereby helps people separate mixed information into manageable clusters. In the context of news, Topic Detection can be seen as an event detection which groups related stories into clusters, each of which stands for a single topic.

Many clustering algorithms have been presented to group topics [1-3]. However, three crucial issues are still not well solved. (1) *The relationship of multiple stories*. The relationship of multiple stories decides the topic each story might belong to. While existing Topic Detection algorithms use the similarity or distance between two stories only, none of them make use of multiple stories’ relationship. Considering multiple stories’ relationship, the corpora could be analyzed more precisely. (2) *The ability to deal with high-dimensional data*. Due to occurrence of long stories, the curse of dimensionality has heavy impact on the performance of topic detection algorithms. (3) *The influence of noise to the clustering result*. Owing to some stories that distribute inhomogeneous in the data model, the detection system may recognize them as noise and cluster them into the wrong topics.

Taken these problems into consideration, we propose a novel framework for topic detection. In this framework, Top-N and PCA are used for dimension reduction, and then a clustering algorithm is employed. We also propose two clustering algorithms for our concerned problem: the Self-tuning Spectral clustering algorithm (SSC) and the Similarity Matrix based Hypergraph Partition algorithm. The experiments show that both improvements are capable of improving the clustering results.

The remainder of this paper is organized as follows: In Section II, we define key concepts and introduce works that are directly related to the ideas presented in this paper. Section III describes our novel approach for clustering topics via improving TFIDF computing formula and using dimensional reduction strategies. In Section IV, we demonstrate the superiority of our approach with comparative empirical results. Finally, in Section V, we present some conclusions and future research directions.

II. RELATER WORK

A. Topic Definition

A topic is defined to be a seminal event or activity, along with all directly related events and activities [4].

Manuscript received December 28, 2010; revised March 1, 2011; accepted March 28, 2011.
Corresponding author: Xinyue Liu

doi:10.4304/jsw.6.12.2407-2415

Thereby, we can deduce that a topic consists of events and activities, both of which are defined in more detail in [5]. A TDT event is defined as something that happens at a specific time and place, along with all the necessary preconditions and unavoidable consequences [5]. For instance, a TDT event may be an earthquake, a conflagration, or an air accident. A TDT activity is a connected series of events with a common focus or purpose that happens in specific places during a given time period [5]. Such a TDT activity might be a presidential election, a social investigation, or a disaster relief effort.

B. Topic Detection

Traditional Topic Detection system comprises three kinds of methods: (1) Single-Pass clustering [1]. In this kind of methods the first story is viewed as the centroid of the topic cluster. When a new story arrives, it will be compared with all other stories by computing the cosine similarity. If the similarity exceeds a threshold, then the story belongs to an old topic, and put it into the most similar cluster; otherwise it will be considered as a new topic and a new cluster will be created. Two disadvantages of this method arise: one is that Single-Pass clustering algorithm depends evidently on the stories sequence; the other is the unbalanced distribution problem. (2) Hierarchical Clustering [2][3]. This kind of methods combines two stories with the highest similarity each time, until all the stories become a single cluster. The complexity of this kind of methods is usually unacceptable. Moreover, the result of clustering is spherical and average， making it more noise sensitive. (3) K-means Clustering [6], in which the K initial cluster center will be selected. The algorithm terminates when the K clustering centers do not change. The performance of this method mainly depends on the selection of the K cluster center, and cluster number K is required.

In recent years, most work focus on proposing better methods on comparison of stories and document representation. A basic incremental TFIDF model is used in [7-10], and new incremental dynamic topic model named topic based TFIDF (T-TFIDF) is proposed in [11]. At the same time, static TFIDF model is widely used [12-16]. If a term frequently appears in different stories, the term weight may be lower than those rarely emerge in different stories in traditional static TFIDF model.

III. OUR APPROACH

First and foremost, preprocess step is provided for the corpus. For each story we tokenize the text, tag the tokens, remove stop words, and get a candidate set of features for its vector-based model. In this paper, a token and its tag are viewed as a feature. If the tokens are spelled the same way but differ in tags, they will be taken as different features. ICTCLAS [17] is used as the tokenizer and tagger, and the stop word list contains 507 words.

A. Term Weighting

Incremental TF-IDF model is used for term weight calculation in this paper.

TABLE I.
MEANINGS OF SYMBOLS

Symbol	Notation
w_{ij}	Weight of term i in story j.
tf_{ij}	Within-story term frequency (TF).
$\log(\frac{n_j}{N}+0.01)$	Inverse document frequency (IDF).
n_j	The number of training story where term j occurs.
N	Size of training corpus.
n_{ij}	Raw count of term i occurrences in story j.
len_j	Total terms number of story j.
len_{avg}	Average terms number in each story.

$$w_{ij} = tf_{ij} \cdot \log(\frac{N}{n_j} + 0.01) . \quad (1)$$

$$tf_{ij} = \frac{n_{ij}}{n_{ij} + 0.5 + 1.5\frac{len_j}{len_{avg}}} . \quad (2)$$

The meanings of symbols emerged in (1) and (2) are explained in TABLE I.

For the purpose of setting the value of w_{ij} between 0 and 1, (1) is normalized. The normalized TFIDF is given as:

$$w_{ij} = \frac{tf_{ij} \cdot \log(\frac{N}{n_j} + 0.01)}{\sqrt{\sum_{j=1}^{N} (tf_{ij})^2 \log^2(\frac{N}{n_j} + 0.01)}} . \quad (3)$$

The terms that we use are preprocessed, and they are representative. A term emerged frequently in different documents may be the representative word of a topic. In case that the weight of this term is lower than others, the Topic Detection algorithm will not cluster this topic perfectly.

As noted in Section II, the inadequacy of current TFIDF model is due to the fact that existing approach for computing IDF do not consider the importance of term's repetition. We now present solutions to this problem and add title information into TFIDF model.

***Mutual information based TFIDF*.** First of all, the importance of mutual information (MI) is shown by a simple example.

15 stories are selected about the topic of State Council from the corpus. The weights of terms are listed in TABLE II. "State Council" should be more important than "Bank" in this topic. On the contrary, the weight of "Bank" is bigger than the weight of "State Council". Hence, traditional TFIDF is not suitable for computing term's weight. In this paper, mutual information is employed for computing term's weight. And the correctness of TFIDF using MI had been proved in [18].

TABLE II.
TFIDF VALUE OF TERMS

Term	TF	DF	IDF	TFIDF
State Council	0.2	1	0.001	0.0002
⋮	⋮	⋮	⋮	⋮
Bank	0.05	3	1.102	0.0551
⋮	⋮	⋮	⋮	⋮

Therefore, mutual information based TFIDF (MI-TFIDF) is

$$w_{ij} = \frac{tf_{ij} \cdot \log(\frac{n_j}{N} + 0.01)}{\sqrt{\sum_{j=1}^{N} (tf_{ij})^2 \log^2(\frac{n_j}{N} + 0.01)}}. \quad (4)$$

In order to give prominence to the key terms and keep down other terms influence, the constant in (4) is taken as 2. The new MI-TFIDF is

$$w_{ij} = \frac{tf_{ij} \cdot \log(\frac{n_j}{N} + 2)}{\sqrt{\sum_{j=1}^{N} (tf_{ij})^2 \log^2(\frac{n_j}{N} + 2)}}. \quad (5)$$

Title based MI-TFIDF. Intuitively, the most important information for a story should appear in the title [19]. The title always describes who, where and what aspect of an event. If the two stories were on the same topic, they would share a part of title information.

We illustrate the above intuition with examples. Terms in **bold face** are reserved in the story's title. And the examples are extracted from the benchmark TDT5.

Story I: A new topic
Docno: XIN_CMN_20030929.0010
Title: (国际)车臣总理食物中毒但已脱离危险
Pro-Processing: **国际**/n **车臣**/ns **总理**/n **食物中毒**/l 脱离/v **危险**/an

Story II: Story on the same topic
Docno: XIN_CMN_20030930.0101
Title: (国际)俄罗斯车臣总理“食物中毒”治愈出院
Pro-Processing: **国际**/n **俄罗斯**/ns **车臣**/ns **总理**/n **食物中毒**/l 治愈/v **出院**/vn

Story III: Story off the topic
Docno: XIN_CMN_20030916.0223
Title: (国际)俄罗斯发生重大交通事故 6 人死亡
Pro-Processing: **国际**/n **俄罗斯**/ns 发生/v 重大/a **交通**/n **事故**/n 死亡/v

Story I is the seed of topic 55044 about “Popov gets food poisoning”. In the title of Story I, we select $Set1$={国际, 车臣, 总理, 食物中毒, 危险} as the feature set. About story II, $Set2$={国际, 俄罗斯, 车臣, 总理, 食物中毒, 出院} is taken as feature set, and $Set3$={国际, 俄罗斯, 交通, 事故} is viewed as title feature set of story III. Analyzing $Set1$, $Set2$ and $Set3$, stories on the same topic always share most of the features, especially the named entities. On the contrary, stories off the topic barely contain the same features in the title.

From the analysis of the example above, it is reasonable to adjust term weighting according to their location. If the term emerges in the title, the weight should be higher than other terms in the context. Hence, the weight functions using title information, namely, title based MI-TFIDF (T-MI-TFIDF) is denoted as followed:

$$w_{ij} = \frac{\alpha \cdot tf_{ij} \cdot \log(\frac{n_j}{N} + 2)}{\sqrt{\sum_{j=1}^{N} (tf_{ij})^2 \log^2(\frac{n_j}{N} + 2)}}. \quad (6)$$

which holds

$$\alpha = \begin{cases} t & \text{if } n_{ij} \in \text{title} \\ 1 & \text{else} \end{cases}$$

where t is an integer greater than 1. In this paper, T-MI-TFIDF is used to construct VSM (Vector Space Model) for the corpus.

B. Top-N and PAC strategies

After preprocessing, the story vector is high-dimensional. Hence, some noise still exists. Two dimensional reduction strategies (i.e. top-N and PCA) are introduced, to remove the noise.

Top-N strategy simply selects the N words with highest term weight for each story from all the terms. However, it is hard to decide the value of N. Therefore, based on story length harmonic average top-N strategy is put forward. Harmonic average is less than arithmetic average when each element is positive number in the set [20]. Besides, harmonic average is to give support to the shorter length of stories, which satisfies the demand of dimensional reduction method. Therefore, harmonic average is selected for dimensional reduction.

The PCA strategy is a usual dimension reduction method [21]. In this paper, we will extract 99% principal components in the experiment.

C. Topic Detection Algorithm

To overcome the drawbacks of existing algorithms, we bring two methods. The one is SSC, and the other is SMHP. We will introduce them in detail in next two sections.

Self-tuning spectral clustering algorithm. Manor et al. proposed a local scale similarity measure $s(i, j) = \exp(-\| x_i - x_j \|^2 / \sigma_i \sigma_j)$ [22] where σ_i is the distance between point x_i and its k-th nearest neighbor. With this local scale similarity measure, we have $\sigma_c > \sigma_b$, and then $\sigma_a \sigma_c > \sigma_a \sigma_b$. That means point a get closer to point c than to point b. The corresponding spectral cluste-

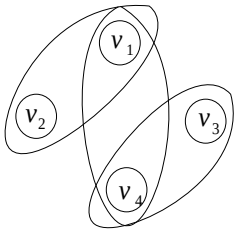


Figure 1. Hypergraph

ring based on this local scale similarity measure is called self-tuning spectral clustering algorithm.

Similarity Matrix based hypergraph partition algorithm. The definition of a hypergraph is given as:

Definition: A hypergraph G can be defined as a pair (V, E), where V is a set of vertices, and E is a set of hyperedges between the vertices. Each hyperedge is a set of vertices: $E \subseteq \left\{\{u, v, \cdots\} \in 2^V\right\}$ [23].

In SMHP algorithm, the most important step is how to transform similarity matrix to hypergraph expression. In this paper, the difference of stories is used and stories are taken as vertices. If the value of difference is smaller than a threshold β, the vertices may be partitioned in the same hyperedge. Following the process of the transformation process is illustrated. For the stories similarity matrix M shown as below,

$$M = \begin{Bmatrix} 1.00 & 0.92 & 0.45 & 0.54 \\ 0.92 & 1.00 & 0.87 & 0.79 \\ 0.45 & 0.87 & 1.00 & 0.36 \\ 0.54 & 0.79 & 0.36 & 1.00 \end{Bmatrix}$$

When the value of β is set to 0.1, the hyperedges are $e_1 = \{v_1, v_2\}$, $e_2 = \{v_3, v_4\}$ and $e_3 = \{v_1, v_4\}$. And the hypergraph is showed in Figure 1.

After transformation, the hypergraph partition tool [24] is called to obtain the clustering result.

Algorithm I shows the exact pseudocode used in this paper.

Algorithm I: SHMP algorithm for Topic Detection

```
For each document d in the corpus do
    Do pre-processing for d
    Compute terms weights
    Flag the length of document d
End for
For each document d do
    Compute harmonic average of all the documents' length
End for
Use H-TOPN and PCA strategies constructing VSM
Compute Similarity Matrix
For each column of the Matrix do
    Descend the column according the value
    Compute the differences diff of two rows
    If diff < β then
        construct a hyperedge
    End if
End for
Use HMETIS toolbox for clustering topics
Output the topic's labels
```

IV. Experimental Results and Analysis

A. Dataset and Evaluation Metrics

In this paper, we used two datasets for our experiment. The one is LDC [19] dataset TDT5, which contains news stories from April to September 2003. TDT5 contains 407,505 stories from source like Xinhua, Associated press, CNN, New York Times, China News Agency, Zaobao News etc. Unlike other TDT corpora, TDT5 does not contain any broadcast news data; all sources are newswire. Only mandarin stories in the collection were considered. In our experiment, we selected 324 stories about 56 topics. The other dataset used in the experiments is gathered from several online news channels. Each news story saved in a separate text file. There are totally 113 stories coming from 21 different topics.

TDT uses a cost function C_{Det}:

$$C_{Det} = C_{Miss} \cdot P_{Miss} \cdot P_{Target} + C_{FA} \cdot P_{FA} \cdot P_{Non-target}. \quad (7)$$

where $P_{Non-target} = 1 - P_{Target}$, and the meanings of symbols in (7) are listed in Table III.

Following a convention in the TDT evaluations, we assign $C_{Miss} = 1.0$, $C_{FA} = 0.1$ and $P_{Target} = 0.2$ .Then C_{Det} is normalized to $(C_{Det})_{Norm}$.

$$(C_{Det})_{Norm} = \frac{C_{Det}}{\min(C_{Miss} \cdot P_{Target}, C_{FA} \cdot P_{Non-target})}. \quad (8)$$

The minimum value of $(C_{Det})_{Norm}$ denoted as $MIN((C_{Det})_{Norm})$, is the optimal value that a system could reach at the best possible threshold [25].

TABLE III.
MEANINGS OF SYMBOLS

Symbol	Notation
C_{Det}	Cost of missing a new story and a false alarm.
C_{Miss}	Cost of missing a story.
P_{Miss}	Probability of missing a story.
P_{Target}	Probability of seeing a new story.
C_{FA}	Cost of a false alarm.
P_{FA}	Probability of a false alarm.
$P_{Non-target}$	Probability of seeing an old story.

TABLE IV.
SUBSYSTEMS OF SYSTEM-1

Name	Strategy
system-1-1	HCA+TFIDF
system-1-2	HCA+T-MI-TFIDF
system-1-3	HCA+T-MI-TFIDF+A-TOPN
system-1-4	HCA+T-MI-TFIDF+H-TOPN
system-1-5	HCA+T-MI-TFIDF+PCA
system-1-6	HCA+T-MI-TFIDF+A-TOPN+PCA
system-1-7	HCA+T-MI-TFIDF+H-TOPN+PCA

TABLE V.
SUBSYSTEMS OF SYSTEM-2

Name	Strategy
system-2-1	KCA+TFIDF
system-2-2	KCA+T-MI-TFIDF
system-2-3	KCA+T-MI-TFIDF+A-TOPN
system-2-4	KCA+T-MI-TFIDF+H-TOPN
system-2-5	KCA+T-MI-TFIDF+PCA
system-2-6	KCA+T-MI-TFIDF+A-TOPN+PCA
system-2-7	KCA+T-MI-TFIDF+H-TOPN+PCA

TABLE VI.
SUBSYSTEMS OF SYSTEM-3

Name	Strategy
system-3-1	SSC+TFIDF
system-3-2	SSC+T-MI-TFIDF
system-3-3	SSC+T-MI-TFIDF+A-TOPN
system-3-4	SSC+T-MI-TFIDF+H-TOPN
system-3-5	SSC+T-MI-TFIDF+PCA
system-3-6	SSC+T-MI-TFIDF+A-TOPN+PCA
system-3-7	SSC+T-MI-TFIDF+H-TOPN+PCA

TABLE VII.
SUBSYSTEMS OF SYSTEM-4

Name	Strategy
system-4-1	SMHP+TFIDF
system-4-2	SMHP+T-MI-TFIDF
system-4-3	SMHP+T-MI-TFIDF+A-TOPN
system-4-4	SMHP+T-MI-TFIDF+H-TOPN
system-4-5	SMHP+T-MI-TFIDF+PCA
system-4-6	SMHP+T-MI-TFIDF+A-TOPN+PCA
system-4-7	SMHP+T-MI-TFIDF+H-TOPN+PCA

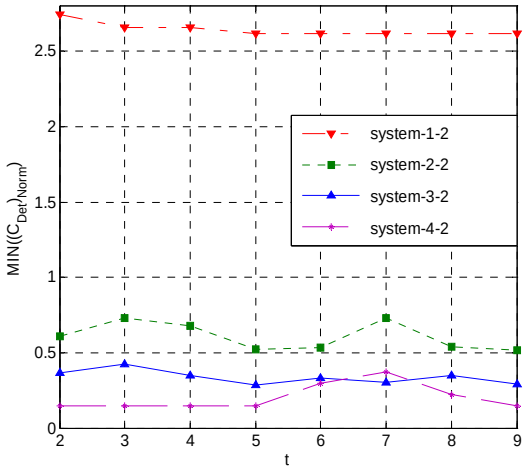


Figure 2. MIN((C_{Det})$_{Norm}$) with different subsystems and *t*

TABLE VIII.
VERIFY T-MI-TFIDF MODEL'S VALIDITY

System	MIN(C_{Det})$_{Norm}$	System	MIN(C_{Det})$_{Norm}$
system-1-1	2.7454	system-1-2	**2.6608**
system-2-1	0.9515	system-2-2	**0.5233**
system-3-1	0.3285	system-3-2	**0.2858**
system-4-1	0.2227	system-4-2	**0.1485**

B. Systems in Experiments

To validate the approaches proposed in the model, we implemented and tested 4 systems including 28 subsystems:

System-1 uses HCA (hierarchical clustering algorithm) and its subsystems listed in TABLE IV. KCA (K-means clustering algorithm) is used in **System-2** and its subsystems listed in TABLE V. **System-3** uses SSC algorithm and its subsystems listed in TABLE VI. SMHP algorithm is used in **System-4** and its subsystems in TABLE VII.

In this paper, the single-pass clustering algorithm is not selected because this method requires the document in order. The corpus collected is out-of-order, so the single-pass clustering algorithm is not tested.

C. Result and Discussion On collected Dataset

Verifying T-MI-TFIDF model's validity. Figure 2 shows the $MIN((C_{Det})_{Norm})$ in subsystems with different *t* values. The lower the $MIN((C_{Det})_{Norm})$ is, the better the system's performance is. From Figure 2, when $t = 5$, all of the four systems can achieve best performance. Thus we will choose $t = 5$ in the next experiments.

TABLE VIII shows the best result of using T-MI-TFIDF model. The best performance of system-1-2 decreases by 8.46% comparing to system-1-1. For system-2-2, the performance decreases by 42.82% compared with system-2-1. And system-3-2's performance is 4.27% lower than system3-1. In similar manner, system-4-2's performance is lower than system-4-1, and the decrement is 7.42%. This phenomenon explains it reasonable to use mutual information and

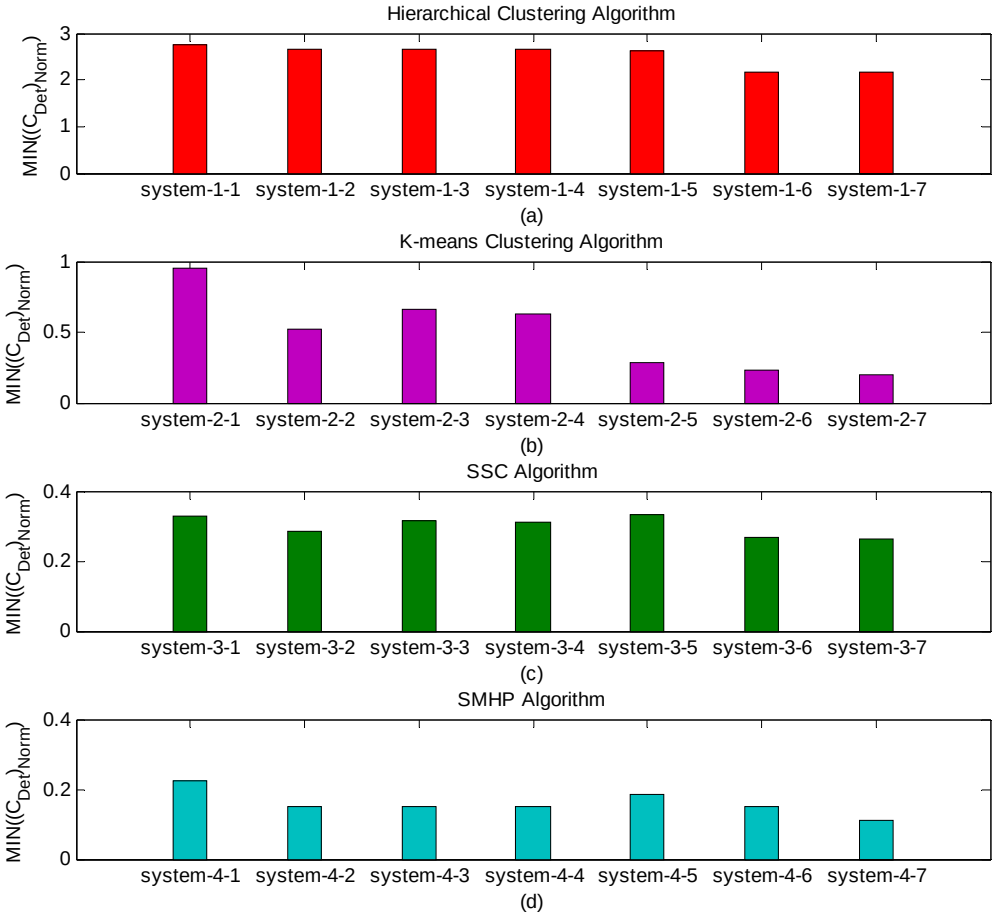


Figure 3. Results of using top-N and PCA strategies

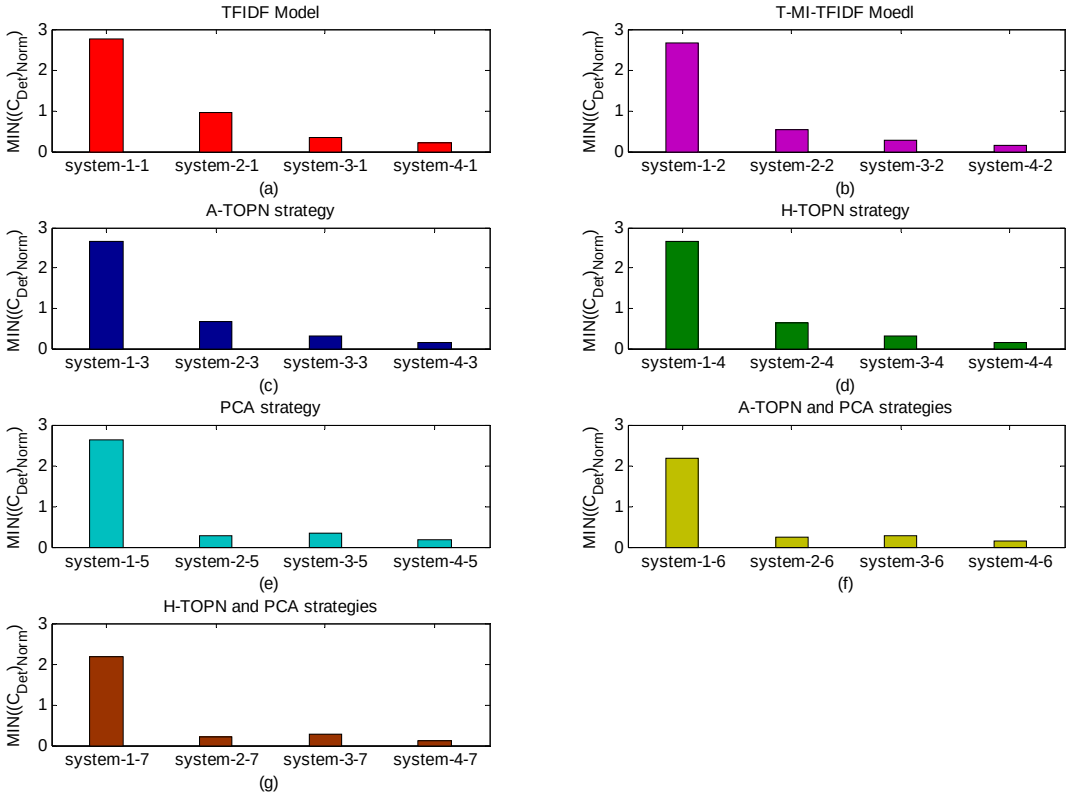


Figure 4. Results of using SMHP algorithm

the title information. Using the mutual information, the weight of features appeared frequently in the stories had been improved; thereby the ability of distinguishing different topics is enhanced. The title information heightened the system's performance as well. The title is the concentration of a story. Hence, T- MI-TFIDF model is more easily to take apart different topics through improving the feature's weight contained in the title.

Testing Top-N and PCA strategies. Figure 3 demonstrates the results of using top-N and PCA strategies for HAC, KCA, SSC and SMHP algorithms.

Through analyzing the results of Figure 3(a) with using HCA, the performance of system-1-5 (using PCA strategy only) is the best comparing to system-1-3 (using A-TOPN strategy only) and system-1-4 (using H-TOPN strategy only). The reason is that some noise had been removed with the PCA strategy. But some of useless information exists still when using H-TOPN strategy, which influences the performance of HCA.

For Figure 3(b) used KCA, system-2-5 shows the best performance compared with system-2-3 (using A-TOPN strategy only) and system-2-4 (using H-TOPN strategy only). The reason is the same as Figure 3(b)'s.

System-3-4 gets the best performance when using SCC comparing to system-3-3 (using A-TOPN strategy only) and system-3-4 (using H-TOPN strategy only) in Figure 3(c), which is different from Figure 3(a) and Figure 3(b). Because self-tuning spectral clustering algorithm is robust to noises. But the dimension decreases obviously with H-TOPN strategy, most of useful information had been restored. Thus the performance of system-3-4 is better than others.

The system-4-4's performance is the best compared with system-4-3 (using A-TOPN strategy only) and system-4-4 (using H-TOPN strategy only) in Figure 3(d). The reason is the same as Figure 3(c). In figure 3, the best performance for each sub graph is the last system (using H-TOPN and PCA strategies). The reason is maintaining the H-TOPN and PCA strategies' advantages and counteracting each other's disadvantages. Thus, the whole system's performance enhanced evidently.

Verifying the validity of SMHP algorithm. Figure 4 shows the results of SMHP algorithm with different strategies.

For each sub graph in Figure 4, the last second system's performance is better than the first two systems. Because SSC algorithm is good at clustering the data distributed inhomogeneous and this method resists the influence of noise strongly. However, other two methods except SMHP algorithm are adequate for clustering low noisy and low-dimensional data.

The best performance for each sub graph is the last system. Through the last paragraph's analysis, a part of reasons had listed. The most important reason is that this method considered the relationship among multiple stories. Just using the relationship, the system recognized the topics easily and exactly.

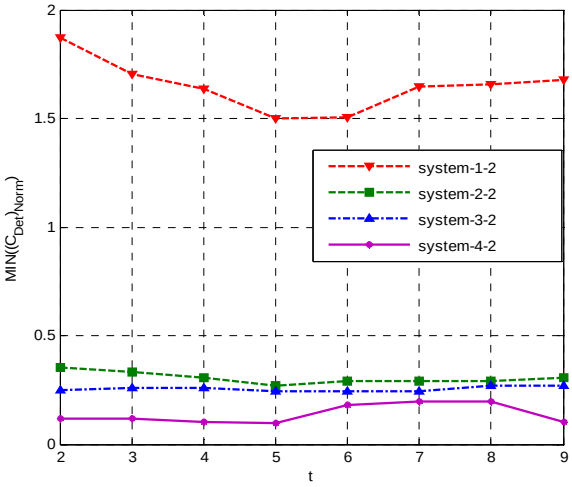


Figure 5. MIN((C_{Det})$_{Norm}$) with different subsystems and t

TABLE IX.
VERIFY T-MI-TFIDF MODEL'S VALIDITY

System	MIN(C_{Det})$_{Norm}$	System	MIN(C_{Det})$_{Norm}$
system-1-1	1.9234	system-1-2	**1.5023**
system-2-1	0.3607	system-2-2	**0.2720**
system-3-1	0.2734	system-3-2	**0.2420**
system-4-1	0.1534	system-4-2	**0.0983**

D. Result and Discussion On TDT5 Dataset

In order to testify the effectiveness of SMHP algorithm, additional experiments are taken on the TDT5 dataset. In Figure 5, when t = 5, all of the four systems can achieve best performance. And using t = 5, T-MI-TFIDF model is better than traditional TFIDF model in TABLE IX. Figure 6 shows the systems can enhance performance using H-TOPN and PCA strategies. From Figure 7, the last system in each sub graph achieves the best performance. It indicates the importance of relationship among multiple stories again.

V. CONCLUSION

In this paper, we have proposed a method for detecting topics in news articles. The algorithm performs well in practice compared to the baseline model. The complexity of SMHP algorithm is $O(|E|)$, which is lower than other algorithms'. It can be seen from the results that mutual information and title information help improving effectiveness of topic detection. Top-N and PCA strategies also improve the performance of clustering result. Our work makes three novel and important contributions:

1. T-MI-TFIDF model for computing the features weight.

2. H-TOPN and PCA strategies' combination for dimensional reduction.

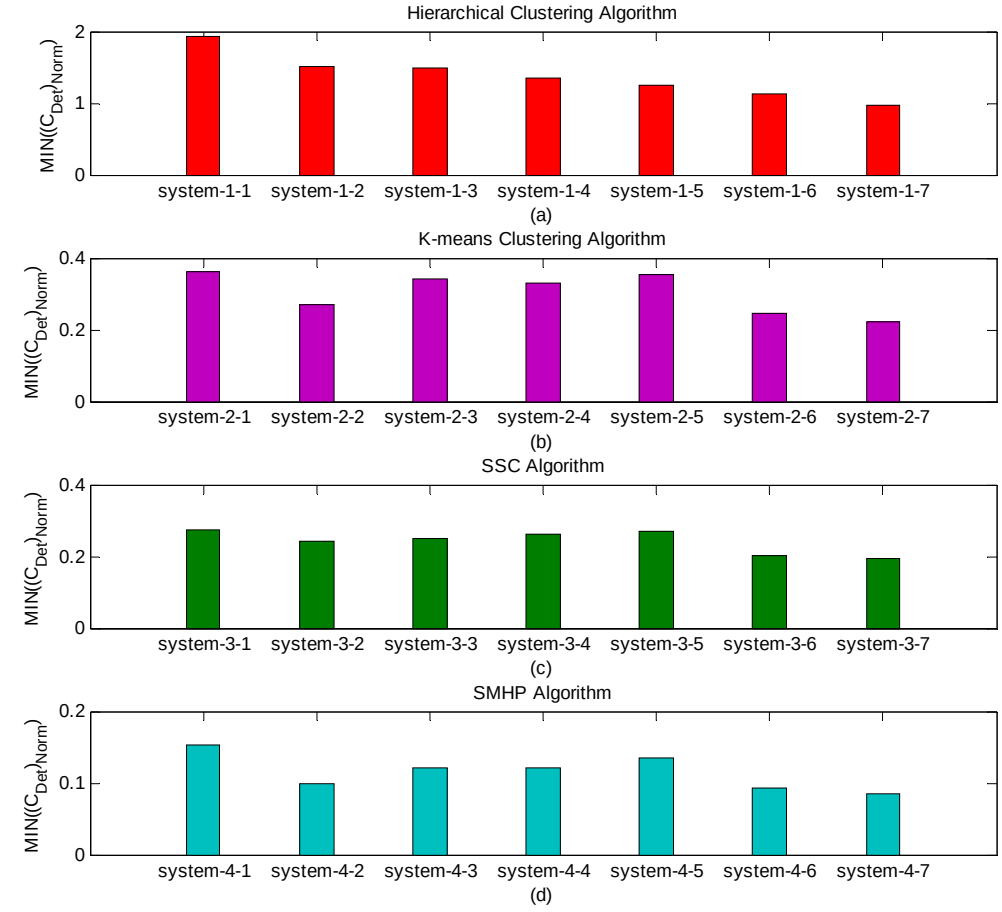


Figure 6. Results of using top-N and PCA strategies

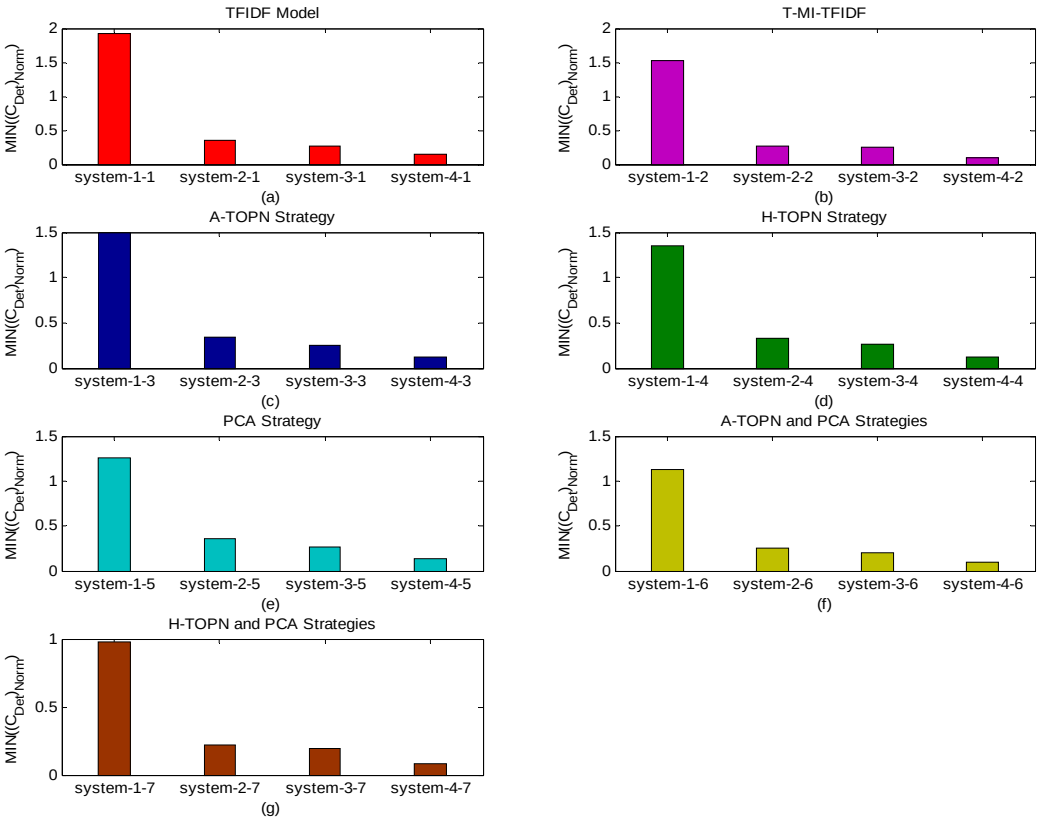


Figure 7. Results of using SMHP algorithm

3. SSC and SMHP algorithms for detecting topics in news stream.

For the Future work, we want to make use of named entities and time information. We also want take advantage of semantic information for distributing different topics.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant No. 60873180 and the Fundamental Research Funds for the Central Universities.

REFERENCES

[1] J. Makkonen, H. Ahonen-Myka, and M. Salmenkivi, "Topic Detection and Tracking with Spatio-Temporal Evidence", *Proceedings of the 25th European Conference on IR Research (ECIR 2003)*, Citeseer, 2003, pp. 251-265.

[2] D. Trieschnigg, W. Kraaij. "TNO Hierarchical Topic Detection Report at TDT 2004". *Topic Detection and Tracking Workshop Report*, 2004.

[3] C. Wartena, R.Brussee, "Topic Detection by Clustering Keywords", *Proceedings of the 2008 19th International Conference on Database and Expert Systems Application*, IEEE Computer Society, 2008, pp. 54-58.

[4] The 2004 Topic Detection and Tracing (TDT'04) Task Definition and Evaluation Plan, http://www.nist.gov/speech/tests/tdt/, 2004.

[5] TDT 2004: Annotation Manual Version 1.2, http://www.nist.gov/speech/tests/tdt/, Aug. 2004.

[6] C. Flynn, J. Dunnion, "Domain-informed Topic Detection", *Proceedings of the 6th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2004)*, Springer, 2004, pp. 617-626.

[7] Y. Yang, T. Pierce, J. Carbonell, "A Study of Retrospective and On-line Event Detection", *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, 1998, pp. 28-36.

[8] T. Brants, F. Chen, A. Farahat, "A System for New Event Detection", *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, 2003, pp. 330-337.

[9] M. Zhu, W. Hu, and O. Wu, "Topic Detection and Tracking for Threaded Discussion Communities", *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, IEEE Press, 2008, pp. 77-83.

[10] K. Zhang, J. Zi, and L.G. Wu, "New event detection based on indexing-tree and named entity", *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, 2007, pp. 215-222.

[11] X. Zhang, T. Wang, "Topic Tracking with Dynamic Topic Model and Topic-based Weighting Method", *Journal of Software*, Vol. 5, No. 5, 2010, pp. 482-489.

[12] J. Allan, H. Jin, M. Rajman, C. Wayne, D. Gildea, and V. Lavrenko, "Topic-based novelty detection", *1999 Summer Workshop at CLSP Final Report*, http://www.clsp.jhu.edu/ws99/tdt, 1999.

[13] L.S. Larkey, F. Feng, M. Connell, and V. Lavrenko, "Language-specific models in multilingual topic tracking", *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, 2004, pp. 402-409.

[14] G. Kumaran, J. Allan, "Text classification and named entities for new event detection", *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, 2004, pp. 297-304.

[15] B. Li, W. Li, and Q. LU, "Topic tracking with time granularity reasoning", *ACM Transactions on Asian Language Information Processing (TALIP)*, ACM, 2006, pp. 388-412.

[16] J. Zeng, S. Zhang, "Incorporating topic transition in topic detection and tracking algorithms", *Expert Systems with Applications*, Elsevier, Vol. 36, No. 1, 2009, pp. 227-232.

[17] Introduction to ICTCLAS, http://sewm.pku.edu.cn/QA/reference/ICTCLAS/FreeICTCLAS/.

[18] S.S. Kou, and Z.J. Wei, "Improved Weighting Formula in Auto Text Classification", Computer Engineering and Design, Vol. 26, No. 6, 2005, pp. 1616-1618.

[19] J. Qiu, L. Liao, and P. Li, "News Recommender System Based on Topic Detection and Tracking", *Rough Sets and Knowledge Technology*, Springer, 2009, pp. 690-697.

[20] W.S. Jevons, *Investigations in Currency and Finance*, Macmillan & Co., London, 1884.

[21] A Tutorial on Principal Components Analysis, http://kybele.psych.cornell.edu/;edelman/Psych-465-Spring-2003/PCA-tutorial.pdf, 2002.

[22] L. Zelnik-Manor , and P. Perona, "Self-tuning spectral clustering", *Advances in neural information processing systems*, Citeseer, Vol. 17, 2004, pp. 1601-1608.

[23] Hypergraph, http://www.itl.nist.gov/div897/sqg/dads/HTML/hypergraph.html.

[24] HMETIS - Hypergraph & Circuit Partitioning, http://glaros.dtc.umn.edu/gkhome/fetch/sw/hmetis/hmetis-1.5.3-WIN32.zip.

[25] W. Zheng, Y. Zhang, Y. Hong, J.L. Fan, and T. Liu, "Topic Tracking Based on Keywords Dependency Profile", Information Retrieval Technology, Springer, 2008, pp. 129-140.

Xinyue Liu received the M.S degree in Computer Science and technology from Northeast Normal University, China, in 2006. She is currently working toward the Ph.D. degree in the School of Computer Science and Technology, Dalian University of Technology, Dalian, China. Her research interests include multimedia information retrieval, web mining and machine learning.

Fenglong Ma was born in Heilongjiang of P.R.China in 1986. He received his B.S. degree in Software Engineering from Dalian University of Technology in 2010, and currently is a master degree candidate in the same place. His major interests lie in topic detection and tracing.

Hongfei Lin received the Ph.D degree from Northeastern University, China. He is a professor in the School of Computer Science and Technology, Dalian University of Technology, Dalian, China. His professional interests lie in the broad area of information retrieval, web mining and machine learning, affective computing.

Query by Humming Systems Using Melody Matching Model Based on the Genetic Algorithm

Jing Qin [1,2]
1.School of Computer Science and Technology, Dalian University of Technology, Dalian, China, 116024
2.College of Information Engineering, Dalian University, Dalian, China, 116622
Email: jqins@yahoo.com.cn

Hongfei Lin Xinyue Liu
School of Computer Science and Technology, Dalian University of Technology, Dalian, China, 116024
Email: hflin@dlut.edu.cn, xyliu@dlut.edu.cn

***Abstract*—Query by humming (QBH) refers to music information retrieval systems where short audio clips of singing or humming act as queries. Melody is considered as the most important feature in the queries and the songs. This paper proposes a QBH system using melody matching model based on the genetic algorithm and improving the ranking result by local sensitive hashing algorithm. An approximate template of the query music is constructed by melody contour aligning algorithm based on GA, which is used to align and correct the input pitch template. The validity of the algorithm is presented by the prototype of QBH system and effects of the algorithm are also shown by the experiment results.**

***Index Terms*—music information retrieval, melody contour representation, audio system**

I. INTRODUCTION

Traditional music retrieval approaches, based on keyword and textual metadata, face serious challenges. If the user is familiar with the name of the song or other significant information to describe the music, retrieval is straightforward. However, if one does not know the title, singer, alternative retrieval methods are necessary. Content-based music information retrieval (MIR) is gaining widespread attention and can be very helpful, since it forsakes the need of keyword. Often, it consists of a form of query-by-example, such as through singing, humming or playing a sample of the piece, as a query to the database.

From the mid-1990s, there is lots of different field research based on the music information retrieval research. In 1995, Asif Ghias[1] first proposed the method based on the Query by Humming, the method is based on up and down of the melody. From then on, most approaches use pitch sequence to represent the query. For retrieval, dynamic programming[2], dynamic time warping (DTW)[3] and hidden Markov models [4]were used. Recently locality sensitive hashing (LSH) [5][6]approaches are used in this field and gets an impressive result. Since 2005, a number of QBH systems have been evaluated in Music Information Retrieval Evaluation eXchange (MIREX)[7][8].

The major challenges for QBH systems include i) queries vary from different people, how to extract audio features which precisely represent music content, ii) how to describe the musical features, and iii) which method to be used for feature matching.

There are three levels of musical feature, physical features, acoustic features and perceptual features. The physical features express audio content on the format of flow media. The acoustic feature mainly includes time and frequency domain features, such as pitch frequency (or fundamental frequence,F0), short-time energy, zero crossing rate, LPC coefficient and MFCC coefficient, etc. They are the most expressive feature of the audio, and are usually used for different phases of speech recognition. Perceptual features reflect people's feelings such as pitch, rhythm, intensity, timbre, etc. And, the perceptual features can usually be extracted by the physical prosperities. Meanwhile, they are used to recognize and judge the music content, such as the emotion that a song wants to express.

Music is a time-dependent sequence of discrete notes, while we still feel that it is a complete entity. Gestalt Theory (GT) [9] is the framework about the psychological phenomena, the mental process and the psychological application. And this theory proofs that human perception form has a hidden rule which is proximal, similar, and continuous, the rule reveals which pattern organization will be perceived on the condition of stimulus feature. Melody is the main perceptive feature of a song, and in 1986, Dowling [10] proofed that melody satisfy the proximity, similarity and continuity of Gestalt theory. Melody contour could be got by pitch tracking approach, and be used to represent a song. Thus, a melody representation model is the main point in QBH system.

Manuscript received December 28, 2010; revised March 1, 2011; accepted March 28, 2011.
Corresponding author: Jing Qin.

doi:10.4304/jsw.6.12.2416-2420

In the previous work, [11] proposed a model based on the melody of the standard pitch template and humming-input pitch template, normalized two templates and got the results though matching. In this paper, the melody model was improved and refined, and the matching algorithm was better to match the final search results than previous work.

The human voice frequency ranges from 50 to 3200HZ, while music pitch usually ranges from 16 to7000HZ (the equivalent of notes C2～a5), there are some differences between them. The humming fundamental frequency is often several times less than the standard one. Furthermore, because each person is not in the same pitch range, it could cause variation. If the normalization is simply used to the input template and standard template, much detail information of the pitch contours neglected, it would lead to the distortion result.

In this paper, we employ a template with standard fundamental frequency range to approximate the input humming template and be matched with the standard template instead. A melody contour alignment algorithm based on GA is suggested, which might be a linear shift to input templates, and seems reserve the detail information rather than using normalization directly. LSH NN search is employed for templates indexing and matching, the final ranked list seems be improve. The system framework is shown in Fig. 1.

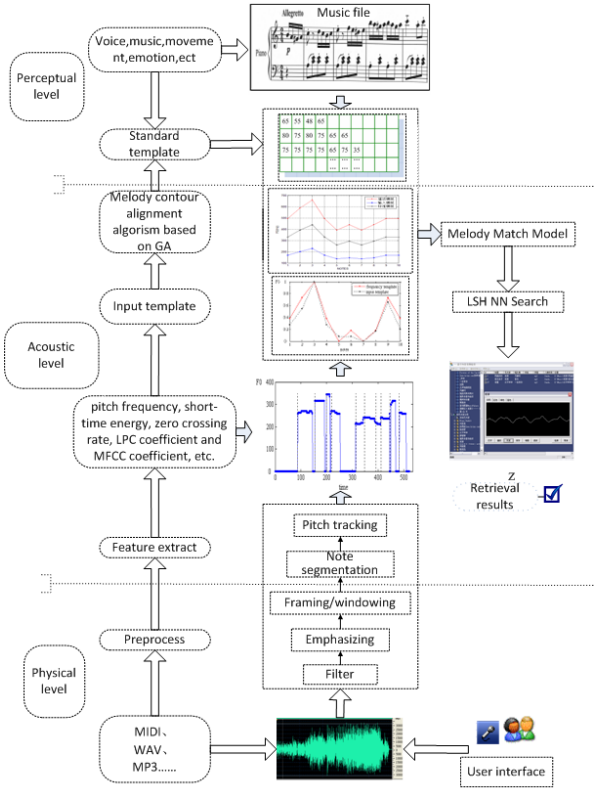


Figure 1. The framework of the proposed QBH system

II. Constructing Melody Match Model

A. Definition

Genetic algorithm [12] as a global optimization algorithm in parallel and global searching capability owns to prominent characteristics. This algorithm does not require the prior knowledge, but can obtain the optimal solution. Therefore, the application using genetic algorithm gain the most similar matching template between the standard pitch contour and the input template, we finally achieved the goal of the template translation.

Melody alignment problem can be defined as follows: Let $P=\{p_1,p_2,\ldots,\ p_i,\ldots,\ p_n\}$ be input template, where p_i represents the certain note, n is the number of notes. We can achieve the approximate template $Q=\{q_1,q_2,\ldots,\ q_i,\ldots,\ q_n\}$ through scaling P template into the range of standard F0 template. In the sequel, we can use similarity measure based on cosine-norm. The goal using cosine-norm is for finding the Q template victor which has the shortest distance with P template.

B. Genetic representation

According to the melody alignment problem, a population of decimal numeral strings, which encoded candidate approximate template with variable length, evolved toward better solutions. A chromosome represented an approximate template, which length varied by the input template. In numbered musical notation, standard pitches distribute into three octaves with 21 notes, the relationship between notes and standard pitches is shown in Table 1.

TABLE1 MAPPING TABLE OF NOTES AND PITCHES

syllable names	Do	Re	Mi	Fa	So	La	Si
F0 (three octave, Hz)	130	146	164	174	196	220	247
	261	293	330	349	392	440	494
	523	587	659	698	784	880	988

Every bit in the population strings was a F0 number in Table1. An input template with a length of *n*, was approximated by a chromosome, which was a decimal numeral strings. An optimized chromosome represented the approximate standard pitch template for an input melody.

For an input template, $P=\{p_1,p_2,\ldots,\ p_i,\ldots,\ p_n\}$, the encoding solution is like that, g_i is a chromosome bit, $i=1,\ldots,n$, $g_i\in\{x\mid x\ is\ \text{standard}\ F0\ \}$, g_i was cording to q_i in approximate template

$Q = \{q_1, q_2, \ldots, q_i, \ldots, q_n\}$, chromosome representation is shown in Table 2. The evolution usually starts from a population of randomly generated individuals, g_i could be one of the standard F0. In each generation, the fitness of every individual in the population was evaluated and modified to form a new population Q. The new population was then used in the next iteration of the algorithm. The algorithm terminates when a satisfactory fitness level has been reached for the best solution, then P and Q are the most closely templates. Because of decimal numeral encoding, the decoding process is simple.

TABLE2 CHROMOSOME REPRESENTATION

q_1	q_2	.....	q_i	.....	q_n
g_1	g_2	.....	g_i	.....	g_n

TABLE3 MELODY CONTOUR ALIGNMENT ALGORISM

Algorithm 1: *Melody contour alignment algorism*

Input: A humming template P;
The max generation number $MAXGEN$;
Output: An approximate template Q;
Description:
1: Choose the initial population of individuals B, randomly produce a number of chromosome;
2: Let S be a chromosome, i be a counter of generations;
3: Evaluate the fitness $F(s) = \cos(P, Q)$ of each individual in population B;
4: Repeat
5: Select the best-fit individuals for reproduction;
6: Breed new individuals through crossover and mutation operations to give birth to offspring;
7: Evaluate the individual fitness of new individuals;
8: Replace least-fit population with new individuals;
9: **until** $i < MAXGEN$;
10: **return** S.

C. Fitness function

Let the input template be $P = \{p_1, p_2, \ldots, p_i, \ldots, p_n\}$, the approximate template be $Q = \{q_1, q_2, \ldots, q_i, \ldots, q_n\}$ through using genetic operation. Each approximate template similarity can be calculated as follow

$$Sim(P,Q) = \cos(P,Q) = \sum_{i=1}^{n} \frac{w_{ip}}{\sqrt{\sum_{k=1}^{n} w_{kp}^{\ 2}}} \times \frac{w_{iq}}{\sqrt{\sum_{k=1}^{n} w_{kq}^{\ 2}}} \quad (1)$$

Through genetic algorithm, the optimal chromosome should own to the maximal fitness function

$$F(s) = Sim(P,Q) = \cos(P,Q). \quad (2)$$

Therefore, cosine-norm measure was used as fitness function, and achieved the optimal approximate template.

D. Melody contour alignment algorism

Process of the melody contour alignment algorism based on GA is described in Table 3.

III. SIMILARITY MATCHING

Euclidean distance is the most widely used similarity measure for time series similarity research. For the input pitch template P and the standard pitch template S, the Euclidean distance is given as (3)

$$D_E(P,S) = \sqrt{\sum_{i=1}^{m} (w_{pi} - w_{si})} \cdot \quad (3)$$

There are lots of advantages of Euclidean distance. For the instance, it can easily calculate and understand, it satisfies the triangle inequality, and supports the multi-dimensional space index.

TABLE4 LSH NN SEARCH ALGORITHM

Algorithm 1: LSH NN search algorithm

Input: An approximate template Q;
A window ω;
Database melody s;
Output: The best similarity song list L;
Description:
1: **For** $i = 1 \ldots ni$ do
2: divide $s(i)$ into several melody fragments $mf(j), j = 1 \ldots mj$;
3: Normalize these melody fragments ;
4: **end for**
5: Index all fragments by LSH;
6: Search approximately k nearest neighbors in the tables of LSH;
7: The candidate melodies are ranked according to the entire database melody occurrence number in k nearest neighbors;
8: Let L be a list of top n candidate songs;
9: return L.

An input audio clip is a part of a song, so we need to get a similarity between a subsequences and a whole sequence which represent pitch information of an entire song. Melody fragment was employed to diverse a song into many subsequences, and constructed an index using locality sensitive hashing by [5].The benefits of LSH against Euclidean distance is that it can get a sublinear time complexity. In our method, we employed the approach above, and made some changes to fit the

melody templates match, the algorithm is described in Table 4.

IV. EXPERIMENTS AND ANALYSIS

A. Music Database

The music database used in this study consisted of 100 pop songs. There are several melody fragments in a song. The input signal which is got from microphone, was sampled on the format of 11.025KHz/8bit/monc, filtered by the bandpass filter with the close frequency of $f_H = 3400Hz$ and $f_L = 60-100Hz$, emphasized by a first-order digital filer $H(Z)=1-\mu^{-1}$, where μ is 0.98. Hamming window is applied to the signal, which is framed with the window size of 128 and with the overlap of 64.

B. Results of GA

The results of GA seem that it is effective in getting approximate templates. For an example, a chinese song named "Tian mi mi", the music score of the first sentence is like "3563121253", the input pitch template is shown in Fig. 2.

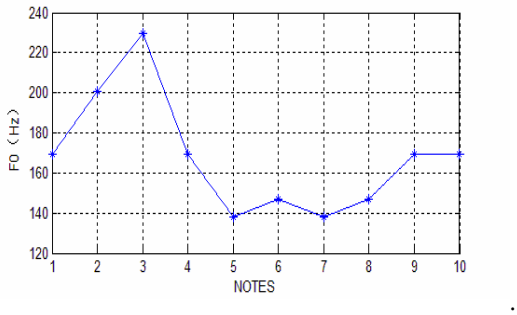


Figure 2. An input template before alignment

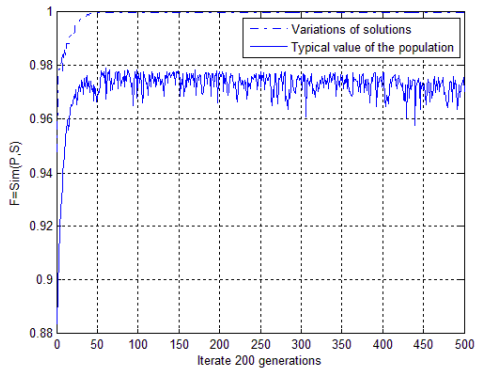


Figure 3. Variations of solutions and typical value of the population size

The original input template would be the input of GA. According to the variable length encoding, the number of notes was another input parameter. In the example, the length of chromosome was 10, population size was 40, generation gap was 0.9, the maximum number of generations was 200, and Fitness-based Reinsertion was used. The result is shown in Fig. 3. The template got a constringency trend after 20 generations, and was steady around 40 computations. The final similarity of the input template and approximate template was 0.9999.

The standard template, input template and approximate template were similar in contour and amplitude, as shown in Fig. 4, even without normalization.

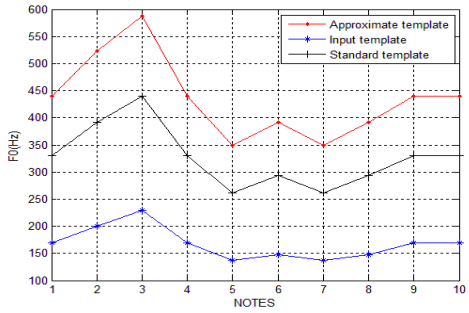


Figure 4. Templates output in melody contour alignment algorism

After normalization, the approximate template coincided with the standard template, which is shown in Fig. 5; it indicates that the difference between each input was eliminated by GA melody contour alignment algorism. It solved a main problem in QBH system as we mentioned above, the best similarity of the approximate template and standard template was a great premise for the matching result.

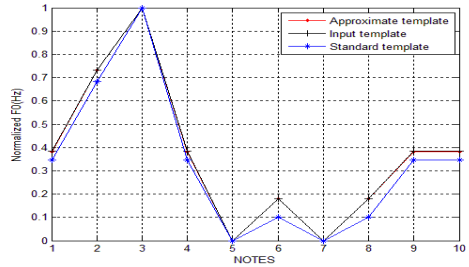


Figure 5. Normalized templates

To test the efficiency of GA, 40 input melodies were involved. The results show that 85% of them got better rank list, the input templates were corrected and the match precision was improved.

C. QBH system search results

Table5 presents the result of 20 piano or other orchestral instruments played audio clips ordered by similarity, under the condition of the great quality input could get Top-3 hit rate of 65%. The process suggested that Top-X hit rate is changed by the number of songs in database, when melody fragments was less, Top-X hit rate was higher. Although Top-3 hit rate was lower, it could make the match melody on the front position, and the similarities of the front melody were close. It indicates that the standard template and input template are similar, but there are many fragments with high similarity in database, they interfere with the results.

TABLE5 INSTRUMENTS MELODY RETRIEVAL RESULTS

Ordinal	1	2-3	4-10	10-20	20-30
Number of melodies	8	5	1	3	1
Top-X hit rate (%)	40	65	70	85	90

To test the humming queries, we randomly analyzed 20 melody fragments from different people which include 10 girls and boys, the results ordered by similarity, as shown in Table 4, the Top-10 hit rate of 50% is lower than instrument fragments, it suggested that the input quality of instrument is better than human, but the system still useful.

TABLE6 HUMMING MELODY RETRIEVAL RESULTS

Ordinal	1	2-3	4-10	10-20	20-30
Number of melodies	2	2	6	3	4
Top-X hit rate (%)	10	0	50	65	85

V. CONCLUSIONS

This paper presented an efficient melody alignment algorithm for solving pitch the differences between different people. We obtained our similarity match algorithm by adapting the general LSH method of [5] to gain a sublinear time complexity. Based on the melody match model, the search process is more fuzzy and effective. After all, the size of database is indeed very small, and we hope that there is room for some improvement. Much work needs to be done to solve interference from high similarity of music fragments in large database.

REFERENCES

[1] Asif Ghias, Jonathan Logan, David Chamberlin, Brian C. Smith, “Query by humming-musical information retrieval in an audio database”, In: Proc of the third ACM international conference on Multimedia .Jan. (1995) 231-236

[2] J.-S. R. Jang, C.-L. Hsu, and H.-R. Lee, “Continuous HMM and its enhancement for singing/humming query retrieval”, in Proc. 6th International Conference on Music Information Retrieval, 2005.

[3] Y. Ohishi, M. Goto, K. Itou, and K. Takeda., “A stochastic representation of the dynamics of sung melody,” in Proc. ISMIR, 2007, pp. 371–372.

[4] J.-S. Roger Jang and Hong-Ru Lee, "A General Framework of Progressive Filtering and Its Application to Query by Singing/Humming", IEEE Transactions on Audio, Speech, and Language Processing, No. 2, Vol. 16, PP. 350-358, Feb 2008.

[5] M. Ryyn¨anen and A. Klapuri, “Query by humming of midi and audio using locality sensitive hashing,” in IEEE International Conference on Acoustics, Speech, and Signal Processing, Las Vegas, Nevada, USA,Apr. 2008, pp. 2249–2252.

[6] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, “Content-Based Music Information Retrieval: Current Directions and Future Challenges”, Proceedings of IEEE, Vol. 96, No. 4, April 2008.

[7] F. Wiering R. Typke and R. C. Veltkamp, “Mirex symbolic melodic similarity and query by singing/humming,” in Intl. Music Information Retrieval Systems Evaluation Laboratory (IMIRSEL), 2006.

[8] A. Ito M. Suzuki, T.J. S. Downie, D. Bryd, T. Crawford, “Ten Years of ISMIR: Reflections on Challenges and Opportunities”, Keynote talk, Kobe, ISMIR 2010.

[9] Nevis, E. (2000) Introduction, in Gestalt therapy: Perspectives and Applications. Edwin Nevis (ed.). Cambridge, MA: Gestalt Press. pp. 3.

[10] W.J. Dowling, “Scale and Contour: Two Components of a Theory of Memory for Melodies”, Psychological Review, lxxxv (1978), 341–54

[11] Jing Qin, Xing-ce Wang, Ming-quan Zhou, Xin-yu Liu, “A novel MIR approach based on dynamic thresholds segmentation and weighted synthesis matching”, in IET Conference on Wireless, Mobile and Sensor Networks 2007 (CCWMSN07), pp.1017–1020

[12] Mitchell, Melanie, (1996), An Introduction to Genetic Algorithms, MIT Press, Cambridge, MA.

Jing Qin received the M.S degree in Computer Science and technology from Beijing Normal University in 2007. She is currently working toward the Ph.D. degree in the School of Computer Science and Technology, Dalian University of Technology, Dalian, China. Her research interests include multimedia information retrieval, music signal processing and machine learning.

Hongfei Lin received the Ph.D degree from Northeastern University, China. He is a professor in the School of Computer Science and Technology, Dalian University of Technology, Dalian, China. His professional interests lie in the broad area of information retrieval, web mining and machine learning, affective computing.

Xinyue Liu is currently working toward the Ph.D. degree in the School of Computer Science and Technology, Dalian University of Technology, Dalian, China. Her research interests include multimedia information retrieval, web mining and machine learning.

Segmenting Webpage with Gomory-Hu Tree Based Clustering

Xinyue Liu[1,2]
1. School of Computer Science and Technology, Dalian University of Technology, Dalian, China
Email: xyliu@dlut.edu.cn

Hongfei Lin[1] and Ye Tian[2]
2. School of Software, Dalian University of Technology, Dalian, China
Email: hflin@dlut.edu.cn

***Abstract*—We propose a novel web page segmentation algorithm based on finding the Gomory-Hu tree in a planar graph. The algorithm firstly distills vision and structure information from a web page to construct a weighted undirected graph, whose vertices are the leaf nodes of the DOM tree and the edges represent the visible position relationship between vertices. Then it partitions the graph with the Gomory-Hu tree based clustering algorithm. Experimental results show that, compared with VIPS and Chakrabarti et al.'s graph theoretic algorithm, our algorithm improves upon the other two with much higher precision and recall, and its running time is far lower than that of Chakrabarti et al.'s graph theoretic algorithm.**

***Index Terms*—Webpage segmentation, DOM tree, Gomory-Hu tree, Planar graph**

I. INTRODUCTION

Nowadays, people rely on search engine to find useful information from the web and web pages are treated as the basic information unit by major commercial search engines. However, they are not atomic units for information search on the Web, since web pages are usually embedded with various kinds of valuable semantic information. Understanding of the structure and the semantics of web pages could significantly improve people's searching experience [1-6]. Web page segmentation, whose aim is to partition a web page into continuous and cohesive regions with unified theme in content and purpose, is an essential task of web page understanding.

There are several applications of web page segmentation. (1) Ranking: segments discriminate between different types of information and can produce more accurate ranking results for different queries with block lever link analysis [1, 2]. (2) Duplicate detection: web pages with identical content information may be presented using different web page layouts, web page segmentation is important to deal with this kind of content duplication. (3) Content Extraction: segments demarcate informative and non-informative content on a web page; removing template noise might also increase classifier performance. Besides web searching and mining tasks, web page segmentation also plays an important role in displaying web pages on small screen devices.

Traditional approaches of web page segmentation usually use heuristic rules [7] or machine learning algorithms either by visual analysis or interpreting the meaning of tag structures. While these approaches might work well on some kind of pages, they still have problems that limit them to be universal applicable. Very recently, Chakrabarti et al.[8] dealt with web page segmentation from a graph-theoretic perspective. They cast the problem as a minimization problem on a weighted directed graph, whose nodes are the DOM tree nodes and the arc-weights express the cost of placing the end points in same/different segments. Chakrabarti et al.'s [8] approach is shown to be superior to previous approaches, however, the underlying graph of this approach is too complex, making the task of solving the minimization problem very difficult. Thus this approach is not efficient and does not suit for real applications.

Our contributions:

(1) We formulate the problem of web page segmentation to the clustering problem on an undirected graph, in which the vertices are leaf nodes of the DOM tree, and the edges represent the neighborhood relationship between vertices when they are rendered on the screen. The graph constructed by both vision and structure information is a planar graph, on which most optimization problems are much easier than those on general graphs.

(2) We deploy the Gomory-Hu tree based algorithm to solve the underlying clustering problem in our formulation. The use of Gomory-Hu tree based algorithm not only because it has theoretical clustering quality guarantees, but also because it can benefit from the graph's planarity to gain very efficient algorithm. However, Gomory-Hu tree based algorithm tends to generate outliers. We use a simple pre-processing technique to remedy this problem.

(3) We evaluate our approach through experiments. The experimental results show that our algorithm

Manuscript received December 28, 2010; revised March 1, 2011; accepted March 28, 2011.
Corresponding author: Xinyue Liu

doi:10.4304/jsw.6.12.2421-2425

improves upon the other two typical algorithms with much higher precision and recall, and its running time is far lower than that of Chakrabarti et al.'s graph theoretic algorithm.

II. Related Work

There is plenty of past work on web page segmentation. Prior work falls into four categories. (1) Heuristics approaches: Cai et al. [7] Proposed the VIPS algorithm that uses rule based heuristics to segment the visual layout of a webpage. Bar-Yossef and Rajagopalan's [9] algorithm identifies template blocks by finding common shingles. Kao et al. [10] presented a webpage segmentation algorithm that relies on content based features. (2) Machine learning approaches: Baluja [5] recast web page segmentation into a machine learning framework using decision tree. Chakrabarti et al.[13] proposed a regularized isotonic regression based algorithm to determine the templates of DOM nodes. (3) Pattern recognition approaches: Xiang et al. [14] define some common patterns beforehand, and then search them in the layout tree.(4) Graph-theoretic approach: Chakrabarti et al. [8] dealt with web page segmentation from a graph-theoretic perspective.

Our approach is also a graph-theoretic approach, thus Chakrabarti et al.'s [8] algorithm is the most closely related one to us. Ours differs from theirs in two ways. Firstly, their algorithm work on a directed graph in which nodes are DOM tree nodes, our algorithm's underlying graph is undirected graph whose vertices are leaf nodes of the DOM tree. What's more, the graph used in our algorithm is planar graph, this property helps to find much more efficient algorithm. Secondly, although the essential tasks of both approaches are graph partition or graph clustering, clustering on directed graph seems much more difficult than clustering undirected graph, and different algorithms are deployed. The Gomory-Hu tree algorithm used in our approach, which is shown to have theoretical quality guarantee, is not applicable to directed graphs.

III. Underlying Graph Construction

Different from Chakrabarti et al.'s approach [8] which uses DOM tree structure information to construct the underlying graph, we use both the DOM tree structure information and the vision information to construct the graph. Initial steps of graph construction are shown as Fig. 1.

A. Vertex Selection

A web page can contain many types of information: content information, vision information, construct information and other inner information, and we segment the web page is to segment the content of the web page, so when choosing vertices we only choose the nodes with real message, such as text, picture and video. In a HTML DOM tree, only leaf nodes contain semantic information, the upper level nodes contain only structural information. Since web page segmentation is to segment semantic information, we use only leaf nodes of the DOM tree as graph vertices. Thus the deduced graph has much smaller number of vertices than Chakrabarti et al.'s graph, and is more informative.

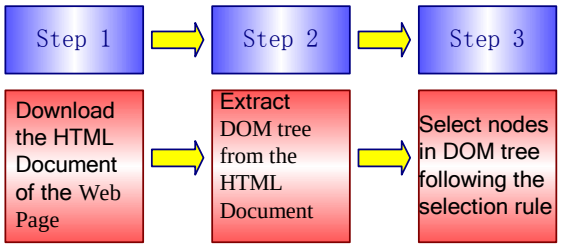


Figure 1. Initial steps of graph construction

We first download the HTML document from the web, second extract the HTML DOM tree, and then choose vertices from the HTML DOM tree depending on the rules. The main principle of vertex selection is choosing the content information, and the rules for vertex selection are showed in Table 1.

TABLE I.
Rules for Vertex Selection

Rules	Content
Rule 1	DOM tree nodes containing text content, link and picture information which appear in the rendered page are vertices of the graph
Rule 2	DOM tree nodes containing web page layout, text font format information are not vertices of the graph
Rule 3	DOM tree nodes containing text content, link and picture information which do not appear in the rendered page are not vertices of the graph
Rule 4	Visual nodes whose width and height on the browser screen is smaller than threshold *a* are not graph vertices
Rule 5	Visual nodes that are non-informative blank regions on the browser screen are not graph vertices

B. Edge Adding

The visual layout information of the web page is very important. We use this information to construct the undirected graph: if two vertices are neighbors on the browser screen, we add an edge between them. Since each such node is a rectangle on the screen, and only the four borders of the rectangle can produce edge, thus the obtained graph is a planar graph.

The edge adding steps are as follows.

(1) Obtain the locate information of DOM tree nodes which are graph vertices: offsetTop, offsetBottom, offsetLeft and offsetRight.

(2) Find neighbors for each vertex. This is done by finding all nodes that are on the top, bottom, left and right of the current, and then select the nearest among them as the neighbor.

(3) Add edges to vertices with the relationship of neighbor, the flowing is an example of adding edges.

Figure 2 shows an example web page: www.qq.com, in which A, B, C, D, E, F, G, H, I and J are vertices of the corresponding graph. Firstly we obtain the reality location of the vertices and then find the neighbors for all vertices. After adding edges for vertices the graph is as Figure 3.

C. Edge Weighting

We use structural information of the DOM tree to add weights to edges of the graphs. Since each vertex in the graph is a leaf node of the DOM tree, which corresponds to a path from the root to the node. We use path similarity to add the edge weight between two vertices.

Given two vertices in the graph V_i and V_j, which corresponds to two nodes of the DOM tree, L_i=D_{i1}, …$D_{im}E_i$ and L_j=D_{j1}, …$D_{jn}E_j$, where D_{ix} and E_i are used to denote the *x* tag and entities corresponding to the nodes *i*. The weight, which is the similarity of the two paths, is defined as:

$$Sim(L_i,L_j)=1-EditDist((D_{i1},\cdots D_{im}),(D_{j1},\cdots D_{jn}))/Max(D_i,D_j) \quad (1)$$

Where EditDist is the edit distance[17], $Max(D_i,D_j)$ is the max length of the path of D_i and D_j. L_i and L_j represent the length of the path.

The edit distance EditDist used to calculate the min time of insert, delete and replace from the source string (s) to the target string (t). It can weigh the similarity of two strings. For example, string s_1="12433",s_2="1233", EditDist(s_1,s_2)=1, because we can insert 4 into 1233 to get 12433. The integrity algorithm can be seen in [17].

Figure 2. An example web page

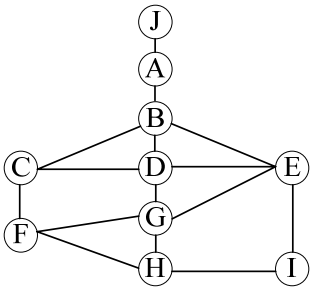


Figure 3. The graph for the web page in Fig. 2

IV. Web Page Segmentation

The task of web page segmentation is essentially to partition or clustering the graph constructed in the above section into groups such that the inner group similarity is maximized while inter group similarity is minimized. Each cluster of the clustering result is a segment of the web page. There are many established clustering algorithms for undirected graphs, we deploy Gomory-Hu tree partition algorithm for both quality and efficiency consideration.

A. Gomory-Hu tree graph partition

Gomory-Hu's[18] partition is a classic graph theoretic algorithm, which partitions the graph by computing minimum cuts and has theoretical quality guarantees[19].

Give an undirected weighted graph $G=(V,E,W)$, where $V=(V_1,V_2,\cdots V_n)$ is the set of vertices, $E=\{e_{ij}=(v_i,v_j)\}$ is the set of edges, and $W=(w_{ij})$ is the set of weights.

Definition 1 (Minimum cut tree or Gomory-Hu tree) [18]: For an undirected weighted graph G, a tree is a minimum cut tree if it follows,

(1) Nodes of the tree are vertices of the graph;

(2) Each edge in the tree has a non-negative weight w_{ij};

(3) For each pair of nodes *s* and *t*, let e_{ij} be the edge on the path from *s* to *t* with the minimum weight, then e_{ij}'s is equal to the capacity of the minimum cut between in *s* and *t* in the graph.

The Gomory-Hu's algorithm constructs the minimum cut tree based on the maximum flow minimum cut theorem. It computes the maximum flow between each pair of vertices to find the minimum cut between them, and constructs the minimum cut tree using these minimum cuts. Gomory-Hu's algorithm requires $O(n)$ maximum flow computations. In a general graph, using the fastest maximum flow algorithm, the computational complexity of Gomory-Hu's algorithm is $O(n^2 m\log(n^2/m))$ [20] where *n* is the number of vertices and m is the number of edges. However, this time bound can be significantly reduced to $O(n^2\log n)$ [21] when the graph is planar. Thus, the use of planar graph in our algorithm can obtain very high efficiency.

For the graph deduced from a web page, in which the vertices represent the visual leaf nodes of the DOM tree, the edges represent neighborhood relationship between them, and edge weight represent similarity between neighbors. Then during the construction of the minimum cut tree, each SupperNode represents continuous region of the web page. The procedure of minimum cut tree construction can be stopped following a best cluttering criteria, thus we can get *k* SupperNodes, i.e., *k* sub-graphs that maps to k blocks in the web page, such that the intra block semantic similarity is maximized, and the inter block semantic similarity is minimized. Thus a good segmentation of the web page is got.

B. Dealing with outliers

Theoretically Gomory-Hu algorithm is an optimal algorithm [18]. However, since it partitions the graph following the minimum cut criteria, it tends to generates outliers, i.e., very small sub-graph or singleton vertices. This is not desirable since in practical applications we usually need the cut to be balanced with some extent. To deal with outliers, we here propose a preprocessing method.

Definition 2 (Outlier): If the sum of edge weights associated to a vertex *i* is smaller than the edge weight sum of its neighbor's by some ratio, i.e,

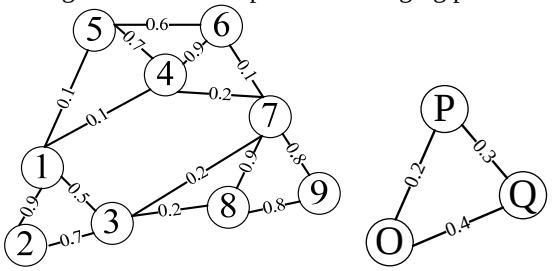
$$\sum_{j\in V} W_{ij} \le \beta \sum_{j,k\in V} W_{jk} \quad (2)$$

Where i is an outlier, and β is a predefined threshold.

To avoid outliers in the final clustering result, potential outliers are merged with their neighbors. The principle for merging is:

$$\max_{i\in S,\, j\in V} W_{ij} \ge \alpha \sum_{i} W_{ij} \quad (3)$$

Where i is a generated vertex merged by a sub set of the vertices; α is a weighted ratio and is set to be 0.1 by experiences. If a vertex j follows the formula, then j is merged into the class i; otherwise, j belongs to another class. Figure 4 is an example of the merging process.



(a)The initial graph (b)The graph after preprocessing

Figure 4. Graph preprocessing

C. Gomory-HuPS Algorithm

The whole Gomory-HuPS algorithm is described as follows:

Gomory-HuPS Algorithm

Input: G is an undirected graph with n vertices;
Output: k parts of G;

```
1   vertex i is one class of G, the number of vertices reduce 1;
2   while(vertex j in all vertices of G){
3       if(i, j satisfy formulae (3)) j is in the class of i, the number of vertices reduce 1;
4       else {
5           if(j satisfy formulae (2)){
6               j is in the class of i, the number of vertices reduce 1;
7               if(the number of vertices is 0) using Gomory-Hu algorithm to get the Gomory-Hu tree of the new graph, taking out the edges ascending, then get k parts of G;
8               else  j is a new class, the number of vertices reduce 1;
9           }
10          else  j is a new class, the number of vertices reduce 1;
11      }
12  }
```

V. EXPERIMENTS

To evaluate the proposed algorithm, which is denoted as Gomory-HuPS, we did experiments on 100 pages with 2726 blocks. The data set is crawled automatically and the blocks are manually segmented. We compared our algorithm with VIPS and Chakrabarti et al.'s graph theoretic approach.

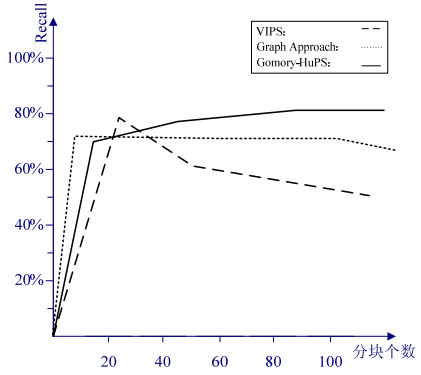


Figure 5. Recall comparison

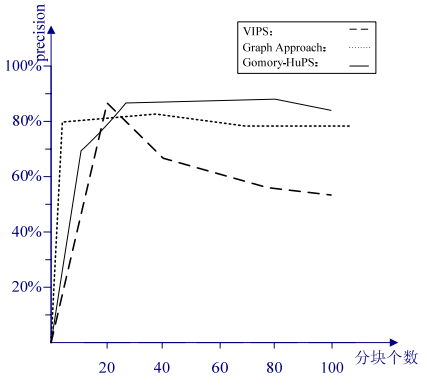


Figure 6. Precision comparison

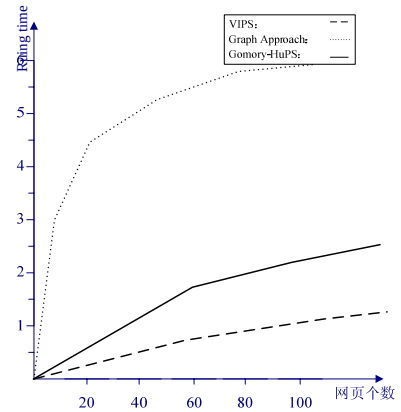


Figure 7. Running time comparison

The experimental results show that our algorithm improves upon the other two with much higher precision and recall, and its running time is far lower than that of Chakrabarti et al.'s graph theoretic algorithm. One reason is VIPS is base on heuristic rules, it might work well on some kind of pages, but it cannot apply to all types of web pages. Gomory-HuPS and Chakrabarti et al.'s graph approach are both base on the graph theory, can apply to any types of web pages. Our approach Gomory-HuPS modifies the graph construct method, uses the Gomory-Hu algorithm to segment the web page, and makes the running time much lower than Chakrabarti et al.'s graph approach.

VI. CONCLUSION

In this paper we have addressed the web page segmentation problem by proposing an algorithm based on Gomory-Hu tree. The algorithm uses vision and structure information from a web page to construct a weighted undirected graph, and then it partitions the graph with the Gomory-Hu tree based clustering algorithm. Since the graph is a planar graph, the algorithm is very efficient. Experimental results show that, our algorithm outperforms both VIPS and Chakrabarti et al.'s algorithm in terms of precision and recall, and is faster than Chakrabarti et al.'s graph theoretic algorithm.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant No. 60873180 and the Fundamental Research Funds for the Central Universities.

REFERENCES

[1] D. Cai, S.P. Yu, J.R. Wen, and W.Y. Ma, "Block Level Web Search", In *Proc. The 27th Annual International ACM SIGIR Conference* (SIGIR'2004), 2004.

[2] D. Cai, X.F. He, J.R. Wen, and W.Y. Ma, "Block Level Link Analysis", In Proc. *The 27th Annual International ACM SIGIR Conference* (SIGIR 2004), 2004.

[3] Z.Q. Nie, J.R. Wen, and W.Y. Ma, "Webpage understanding: beyond page-level search", *ACM SIGMOD Record*. 2009, 37(4):48-54.

[4] J. Zhu, Z.Q. Nie, J.R. Wen, B. Zhang, and H.W. Hon, "Webpage Understanding: an Integrated Approach", In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (KDD07), 2007, 903-912.

[5] S. Baluja, "Browsing on Small Screens: Recasting Web-Page Segmentation into an Efficient Machine Learning Framework", In *Proc. The 15th International Conference on World Wide Web* (WWW2006), 2006.

[6] X.Qi and B.D.Davison, "Web Page Classification: Features and Algorithms", *ACM Computing Survey*, 2009, 41(2): Article 12.

[7] D. Cai, S.P. Yu, J.R. Wen, and W.Y. Ma, "VIPS: a Vision-based Page Segmentation Algorithm", *Technical Report* MSR-TR-2003-79, 2003.

[8] D. Chakrabarti, R. Kumar and K. Punera, "A Graph-Theoretic Approach to Webpage Segmentation", *The 17th International Conference on World Wide We*b (WWW 2008), 2008.

[9] Z. Bar-Yossef, and S. Rajagopalan, "Template detection via data mining and its applications", In *Proc. 11th International Conference on World Wide Web* (WWW2002), 2002, 580–591.

[10] H.Y.Kao, J.M. Hoand and M.S. Chen, "WISDOM: Web intra-page informative structure mining based on document object model", *IEEE Transactions on Knowledge and Data Engineering*, 2005, 17(5):614–627.

[11] Y. Liu, Q. Wang and Q.X. Wang, "A Heuristic Approach for Topical Information Extraction from News Pages", *The 7th International Conference on Web Information Systems Engineering* (WISE2006), 2006.

[12] D. Debnath, S. Mitra, N. Pal and C.L. Giles, "Automatic Identification of Informative Sections of Web Pages", *IEEE Transactions on Knowledge and Data Engineering*, 2005, 17(9): 1233-1246.

[13] D. Chakrabarti, R. Kumar and K. Punera, "Page-level template detection via isotonic smoothing", In *Proc. The 16th International Conference on World Wide Web* (WWW2007), 2007, 61-70.

[14] P.F,Xiang, X. Yang and Y.S. Shi, "Web Page Segmentation based on Gestalt Theory", In *Proc. IEEE International Conference on Multimedia and Expo*, 2007, 2253-2256.

[15] H. Srinivasan , R. Rupesh and M. Amit, "Web Page Layout Optimization Using Section Importance", *The 17th International Conference on World Wide We*b (WWW 2008), 2008.

[16] Sandip Debnath, Prasenjit Mitra, Nirmal Pal, and C. Lee Giles. Automatic Identification of Informative Sections of Web Pages. IEEE Computer Society.2005.

[17] D. Gusfield, *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, 1997.

[18] R.E. Gomory and T.C. Hu, "Muti-Terminal Network Flows", *Journal of the SIAM*, 1961, 9: 551-570.

[19] A.V. Goldberg and R.E. Tarjan, "A new approach to the maximum-flow problem", *Journal of the ACM*, 1988, 35(4): 921-940.

[20] G. Borradaile, and P. Klein, "An O(nlogn) Algorithm for Maximum st-Flow in a Directed Planar Graph", *Journal of the ACM*, 2009, 56(2): Article 9.

[21] G.W. Flake, R.E. Tarjan and K. Tsioutsiouliklis, "Graph Clustering and Minimum Cut Trees", *Internet Mathematics*, 2002, 1(4): 385-408.

Xinyue Liu received the M.S degree in Computer Science and technology from Northeast Normal University, China, in 2006. She is currently working toward the Ph.D. degree in the School of Computer Science and Technology, Dalian University of Technology, Dalian, China. Her research interests include multimedia information retrieval, web mining and machine learning.

Hongfei Lin received the Ph.D degree from Northeastern University, China. He is a professor in the School of Computer Science and Technology, Dalian University of Technology, Dalian, China. His professional interests lie in the broad area of information retrieval, web mining and machine learning, affective computing.

Ye Tian received her M.S. degree in Software Engineering from Dalian University of Technology in 2008. Her major interests lie in web page segmentation.

A Two-Dimension XML Encoding Method based on Variable Length Binary Code

Jie Chen[1], Wenxin Liang[2*], Haruo Yokota[3]
[1,2] School of Software, Dalian University of Technology
[3] Tokyo Institute of Technology
Email: [1] chen@mail.dlut.edu.cn, [2] wxliang@dlut.edu.cn, [3] yokota@cs.titech.ac.jp

Abstract— **Recently, the researchers have proposed a number of labeling schemes. In these labeling schemes, the approach which can extract structural information between nodes and process query efficiently is more outstanding. However, most of these labeling schemes do not well support update operations. To achieve update-friendly operations, some of the methods keep intervals between labeling numbers, but it requires whole relabeling when the intervals are used up. Several labeling schemes support dynamic XML documents, but most of these labeling schemes allow only leaf node insertions. OrdPathX supports both leaf node insertions and internal node insertions. Inspired by the method of inserting internal nodes of OrdPathX and extending the C-DO-VLEI code, in this paper we propose two dimensions VLEI code. We discuss how this labeling scheme labels nodes and how we can get the structural information of nodes from their labels. We design experiments to evaluate the efficiency of producing labels, the storage consumption and the querying performance of two dimensions VLEI code we proposed, and compare those with the OrdPathX.**

Index Terms— **XML, Labeling Scheme, Performance Evaluation, Internal Node Insertion**

I. INTRODUCTION

Recently, XML has become a standard language for data representation and exchange over the Internet, and is more and more widely used in various applications. An XML document can be represented as a nested tree. Figure 1 shows an example XML document, which is used throughout this paper. Figure 2 is the corresponding XML tree. According to the different structures in the XML tree, nodes can be classified into three kinds of types: 1) element node, 2) attribute node and 3) text node.

In the past few years, researchers have proposed many labeling schemes. These labeling schemes can be divided into four categories, namely, sub-tree labeling, prefix-based labeling, multiplicative labeling and hybrid labeling [3], and researchers analyze how each approach works, as well as its advantages and disadvantages [3], such as Tree Traversal Order [2] and Dewey Order [9]. Tree traversal order is one of the interval encoding of sub-tree labeling. In this scheme, each node is labeled with a pair of unique integers consisting of preorder and postorder traversal sequences. This label scheme can determine the Ancestor-Descendant (A-D) relationship easily, but the Parent-Child (P-C) relationship can not be

```
<BOOK ISBN ="1-55860-438-3 ">
   <SECTION>
      <TITLE> Bad Bugs</TITLE>
           Nobody loves bad bugs.
      <FIGURE CAPTION ="Sample bug" />
   </SECTION>
   <SECTION>
      <TITLE> Tree Frogs </TITLE>
           All right-thinking people.
      <BOLD> love</BOLD>
         tree frogs.
   </SECTION>
</BOOK>
```

Figure 1. Example XML document.

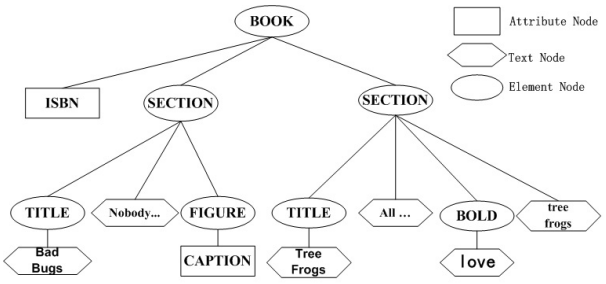


Figure 2. Example XML tree.

determined directly. Dewey Order uses delimiter "." to separate the label of an ancestor node at each level of the tree. In this method, each label of an ancestor node is a prefix of its descendants, the P-C, A-D and sibling relations can be determined by comparing the labels of two nodes. However, both of these two methods are unsuitable and inefficient for a dynamic XML document, because when a new node is inserted into the tree using these two methods, a large number of nodes need to be re-labeled, in order to remain the structural information.

To reduce the cost of insertion, in practice, the update-friendly labeling scheme is more useful. Vector Order [11], VLEI (Variable Length Endless Insertion) code [4], ORDPATH [8] and Extended Preorder Traversal [6] are all update-friendly labeling schemes. [12] compares Vector Order with some other update-friendly labeling schemes. In this paper, we only focus on the methods combined with Dewey Order. ORDPATH is one

*Corresponding author.

doi:10.4304/jsw.6.12.2426-2433

of the famous update-friendly labeling schemes based on the prefix-based labeling scheme Dewey Order. ORDPATH uses the variable-length bit string to represent labels. ORDPATH is conceptually similar to the Dewey Order described in [9]. In ORDPATH, only positive and odd numbers are assigned for the initial labeling, such as the code of 1.1.1.3 and 1.5. Even and negative numbers can be preserved for future insertions. In the actual encoding, labels are represented as the compressed ORDPATH format. Because in practical applications, many large XML documents need to be handled, therefore, it is necessary to ensure enough storage for extracting node's structural information and update operations. (DO-VLEI) code [**?**] and ORDPATH [**?**] have been proposed. The DO-VLEI labeling method inherits features of the Dewey Order method, but reduces the update cost for insertion operations. It uses the VLEI code [4] for expressing the sibling order by a unique magnitude relationship, which enables the unlimited insertion of new nodes with no relabeling of other nodes being required. Experiments in [7] indicates that the DO-VLEI code outperforms the ORDPATH in both structural information extraction and storage consumption.

However these update-friendly labeling schemes are allow only leaf node insertions, if the intermediate node is inserted, the methods cannot guarantee there is no need to recalculate the labels of large number of nodes. In practical applications, it is no doubt that the situations of inserting intermediate nodes are possible, therefore [1] proposed a two-dimensional encoding method called OrdPathX. OrdPathX is a two-dimensional labeling scheme based on the "careting-in" technique of ORDPATH. It can handle both internal and leaf node insertions efficiently. Because one-dimensional VLEI code outperforms ORDPATH, we want to design a two-dimensional VLEI code which support intermediate node insertion based on one-dimensional VLEI code. According to the analysis of advantages and disadvantages of various labeling schemes, we design experiments to compare the two-dimensional VLEI code with OrdPathX, in order to prove whether the two-dimensional VLEI code outperforms OrdPathX. Experiments show that the two-dimensional VLEI code we proposed outperforms OrdPathX in both structural information extraction and storage consumption.

In this paper, we mainly introduce the two-dimensional VLEI code and discuss how the proposed labeling scheme labels nodes and how to handle the issue of re-labeling when inserting intermediate nodes in the vertical dimension. Then we compare the two-dimensional VLEI code with OrdPathX in the following three respects:

1) The label construction speed.
2) Whether we can get node's information from the generating label efficiently, the information concludes node depth information, the node labels and node names of the parent node and the ancestor node.
3) The average length of the labels, which determines whether the labeling scheme saves storage space.

Experimental results show that the two-dimensional VLEI code outperforms OrdPathX in these three respects, it is mainly because the compressed two-dimensional VLEI code's length is shorter than the compressed OrdPathX, thus reducing storage consumption, and compress efficiently. Because in OrdPathX when compressing and decoding each component must refer to the prefix schema and even-numbered and negative integer component values are reserved for later insertions, so it is slower than the two-dimensional VLEI code which does not use the prefix schema in the label construction and the structural information queries.

The remainder of the paper is organized as follows: Section II introduces related researches. Section III describes our proposed two-dimensional VLEI code. Section IV shows the method for extracting useful structural information from the two-dimensional VLEI code. Section V describes the experiments and evaluation of our proposed labeling method. Section VI concludes the paper.

II. Related Work

Dewey Order is a simple prefix-based labeling scheme, and many other labeling schemes are based on Dewey Order, Dewey Order [9] uses "." delimiter to separate its parent label and its own label, It is defined as follows:
1. The label of the root node $C_{root} = 1$.
2. The label of a non-root node C= $C_{parent}.C_{child}$, where C_{parent} denotes the label of its parent node, C_{child} denotes the order of the node in the sibling node.

We can get node's depth information, the label of its parent node and ancestor nodes and sibling (preceding or following) relationship from the label. But Dewey Order is not an update-friendly labeling scheme, therefore, researchers proposed ORDPATH [8] and DO-VLEI [4], and both of the two schemes are based on the Prefix-based labeling scheme Dewey Order. The two methods do not have re-labeling problem after insertion. We will introduce the two methods next.

A. ORDPATH

ORDPATH is implemented in Microsoft®SQL $Server^{TM}$ 2005, and is used in the added attribute HierarchyID in the latest release of Microsoft®SQL $Server^{TM}$ 2008. ORDPATH is an update-friendly labeling scheme, based on Dewey Order, in Dewey Order, node's label is made from the label of the parent node, a "." delimiter and a brother codes. In ORDPATH, it is similar to Dewey Order during the initial labeling, but only positive, odd integers are assigned. Figure 3 is an XML tree labeling by ORDPATH, even number and negative integer component values are reserved for further node insertions. For example, when inserting a node between nodes labeled "1.3.1" and "1.3.3", the new node will be labeled by "1.3.2.1", in which "2" is a placeholder not increasing the depth of node, is assigned to the node. The node of "1.3.2.3" is inserted between

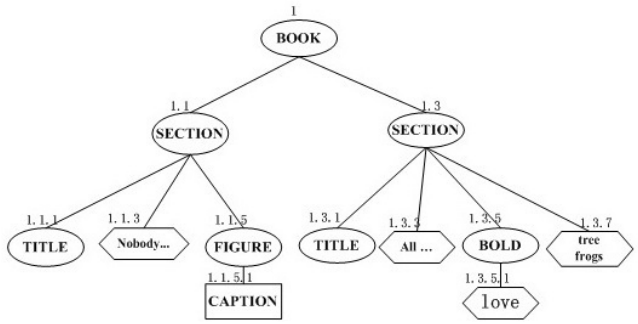


Figure 3. Labeling by ORDPATH.

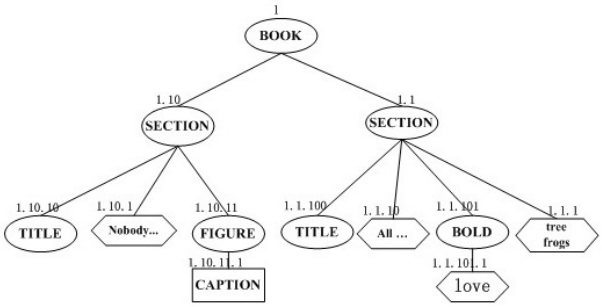


Figure 4. Labeling by DO-VLEI code.

nodes labeled "1.3.2.1" and "1.3.3". ORDPATH can support update operations without relabeling. If given the label of a node, its parent nodes, ancestor nodes and depth information can be got, and the order of nodes also can be quickly got by comparing the labels.

Note that the dots in the label are just for readers' easy understanding. In the actual encoding, we use the variable-length bit string to represent labels, called C-ORDPATH. In the C-ORDPATH, label is expressed as successive variable-length Li/Oi bit strings, and each Li/Oi is used to represent an integer. according to a prefix schema [8]. Li represents the length of Oi, and it is a prefix-free encoding. Oi represents the value of the component. For example, "1.3.1" is compressed into 0110101 according to the prefix schema in [8]. Because 1 is represented as 01(01 is Li, and it represents the length of Oi is 0, so 1 does not have Oi), and 3 is represented as 101(10 is Li, and it represents the length of Oi is 1, so the Oi of 3 is 1 according to the prefix schema), "1.3.1" is represented as 0110101.

B. DO-VLEI Code

DO-VLEI [4] is the labeling scheme combining VLEI code with Dewey Order. We will introduce VLEI code first.

VLEI code is a variable-length bit string starting with 1 and is composed by 0, 1, the relationship between codes satisfies the following definition.

Definition 1: the relationship between VLEI codes: v is a VLEI code, and the following condition is satisfied: $v \cdot 0 \cdot \{0|1\}* < v < v \cdot 1 \cdot \{0|1\}*$

The new code will be smaller than the original label if 0 is added behind a VLEI code. The new code will be larger than the original label if 1 is added behind a VLEI code. So that there is no need to change other nodes' labels when inserting a new node and label-generating is also easy. The method combining VLEI code with Dewey Order is DO-VLEI code. It is defined as follows:

1. The DO-VLEI code of the root node $C_{root} = 1$.

2. The DO-VLEI code of a non-root node C= $C_{parent}.C_{child}$, where C_{parent} denotes the DO-VLEI code of its parent and C_{child} denotes the order of the node in the sibling node.

Figure 4 is an example XML tree labeling by DO-VLEI code. Each component in DO-VLEI code is a VLEI code, separated by ".". When generating each component in DO-VLEI code, we have to use an algorithm [5] that mapping a natural number to VLEI code. The algorithm is given for generating each component of node when labeling the label of each node during the initial XML tree load.

We need to use the algorithm in [5] when inserting nodes. The algorithm is about how to decide the VLEI code when inserting nodes. When inserting a node v between two sibling nodes the VLEI code are v_l and v_r (v_l is the left sibling, v_r is the right sibling, $v_l < v_r$) respectively, then it's VLEI code is decided like this: If the length of v_l is greater than the length of v_r, the VLEI code of v is v_l1. Otherwise, if the length of v_l is less than or equal to the length of v_r, the label of v is v_r0. So the VLEI code can ensure inserting nodes without changing other nodes' labels and the insertion label is unique. In the DO-VLEI code, insert operation effects only the last component, the head components only need to inherit the DO-VLEI code of the parent node.

C. C-DO-VLEI Code

The same as ORDPATH, in order to save storage space, in the actual storing, we need to compress DO-VLEI code, and the encoding scheme called C-DO-VLEI code [7]. A DO-VLEI code is composed of three elements: ".1", "0" and "1", at most 2 bits can represents three different types of elements. ".1", "0" and "1" are represented by 10, 0 and 11 respectively when compressing DO-VLEI code. This compressed code called C-DO-VLEI Code. For example, 1.10 .11 is compressed into 111001011. 0, 10, and 11 are prefix codes, so that C-DO-VLEI codes can be uniquely decoded into the original DO-VLEI codes. Scan C-DO-VLEI Code front to back when decoding, look at the number of consecutive 1s before the 0. If there is an odd number of consecutive 1s appearing before 0, the last two bits 10 represent .1.Otherwise, if there is an even number of consecutive 1s before 0, all of 1s are decoded into 1, and the final "0" is decoded into 0. We can see that in compressing and decompressing DO-VLEI code is easier than ORDPATH, it doesn't require any prefix schema, and it has already been proved in [7] that C-DO-VLEI code outperforms C-ORDPATH in the storage consumption and query performance.

D. OrdPathX

Both ORDPATH and DO-VLEI Code are update-friendly labeling schemes. It does not require relabeling for other nodes when inserting leaf node, but they cannot handle internal node insertion effectively. OrdPathX [1] can handle both internal and leaf node insertions efficiently. In OrdPathX, the label consists of an Augmented OrdPath(AO), possibly followed by a Parent Height (PH):$OrdPathX = AO.PH$, where an AO is a sequence of chunks and each chunk is consisted of zero or more even components followed by an odd component. Between each pair of consecutive chunks there may exist an Incremental Height (IH). OrdPathX is inspired by the "careting-in" technique of ORDPATH. In OrdPathX, it is the same as ORDPATH during the initial load, and all of the labels of nodes don't have IH and PH. The label of node v is $L.C$, and the label of its parent node v' is L. Now we are to insert nodes u_l and u_2 between them. First, consider the insertion of u_1, the label of u_1 is $L.(1).C$ where "(1)" is an IH. Moreover, v relabeled as $L.C.[1]$ where "[1]" is the PH of v's label. It indicates that v has a parent-insertion node above it. We can see that if a node's label has PH, a node has been inserted as its parent, and its PH value is equal to the IH value of its parent node's label. Then insert u_2 as the new parent of u_1. The label of u_2 is $L.(3).C$ where (3) is the IH value of u_2, so u_1 should be relabeled as $L.(1).C.[3]$.

In OrdPathX, the use of dots and brackets in the labels are just for readers' easy reading. In the actual encoding, label is represented as the form of variable-length bit string. Each component is encoded using the $IiLiOi$ format where Ii is the component type indicating whether the component is an ORDPATH, IH or PH. Ii uses a fixed size of 2 bits to represent since there are three different types of label components. The compression of $LiOi$ is the same as ORDPATH. We can use 01 to represent IH, 10 to represent PH and 00 to represent ORDPATH. We can see that in OrdPathX both compressing and decompressing need to refer to the prefix schema, and the selection of the prefix schema is also important.

III. 2-D VLEI CODE

In this section, we propose 2-D VLEI Code based on DO-VLEI Code [4]. It not only supports leaf node insertion but also supports internal node insertion.

A. Labeling Method

Because DO-VLEI Code and ORDPATH are both based on Dewey Order, they are very similar in leaf node insertion. The 2-D labeling scheme OrdPathX based on ORDPATH can handle internal node insertion, and can extract the structural relationship between nodes from labels. Therefore, according to the method that OrdPathX handles internal node insertion, we proposed 2-D VLEI code based on one-dimensional VLEI code to achieve internal node insertion efficiently.

The label in OrdPathX consists of an Augmented OrdPath(AO), possibly followed by a Parent Height (PH):

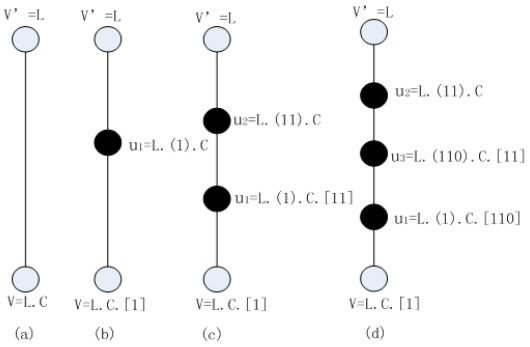


Figure 5. An example of internal node insertion by 2-D VLEI code.

$OrdPathX = AO.PH$. The 2-D VLEI code we proposed is based on OrdPathX, so there are IH and PH in 2-D VLEI code, but IH and PH here are components, not chunks.

B. Internal Node Insertions

In 2-D VLEI code, all the labels is the same as DO-VLEI Code during the initial load, and all of the labels of nodes don't have IH and PH, and use the same method inserting leaf nodes. Therefore, this section focuses on the method of internal node insertion.

The label of node v is $L.C$, and the label of its parent node v' is L (Figure 5(a)). Now we are to insert nodes u_1, u_2, and u_3 between them. First, consider the insertion of u_1, We label u_1 as $L.(1).C$ where "(1)" is an IH, as shown in Figure 5(b). The value of IH is the first VLEI code 1 according to the insertion method of VLEI code, and then v is relabeled as $L.C.[1]$ where $[1]$ is a PH. Next, insert u_2 between u_1 and v', we label u_2 as $L.(11).C$ and also re-label u_1 as $L.(1).C.[11]$, as shown in Figure 5(c). Finally we insert node u_3 between u_1 and u_2. Because the IH of u_1 and u_2 are (1) and (11) respectively, the IH of u_3 is the value (110) which is bigger than (1) and smaller than (11) according to the insertion method of VLEI code. We label u_3 as $L.(110).C.[11]$ and re-label u_1 as $L.(1).C.[110]$, as shown in Figure 5(d). And this IH is the Significant Incremental Height (SIH) because the component's presence implies that the node is a parent-insertion, similar as the conception of SIH in [1].

There is a special situation of leaf node insertion. If v is a parent-insertion node, we cannot handle the insertion as the normal method, because v has a child, say w, which was present before v was inserted. The label of v is $L.(IH).C.[PH](PH$ can be empty). If u is inserted on the left of w, we label u as $L.(IH).C.LS$ (Left Sibling, the new node is the left brother of the present node w). An example is shown in Figure 6. If we insert a leaf node as the child of node $L.(1).1$ on the left of the node whose label is $L.1.[1]$, the new node's label is $L.(1).1.1$. If u is inserted on the right of w, u is labeled as $L.(IH).C.D$. In Figure 6, the new node is labeled as $L.(1).1.1$. Assuming that the new child node is v', if

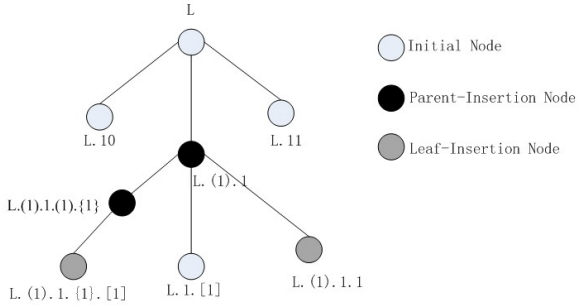


Figure 6. A special example of leaf node insertion.

we are to insert node u between v and v', the new node is labeled as $L.(IH).C.(IH).LS$ or $L.(IH).C.(IH).D$, the PH value of v is equal to the last IH value of u. If we insert a node between the node $L.(1).1.1$and the node $L.(1).1$, the new node is labeled as $L.(1).1.(1).1$, the leaf node is relabeled as $L.(1).1.1.[1]$,as shown in Figure 6.

C. Structural Information

We can get the inter-node relationships between two nodes from their 2-D VLEI code (namely, P-C, A-D and siblings (preceding sibling and following sibling)):

P-C relation: Given two nodes u and v, if we want to verify whether u is v's parent node, we just see if the $L.(IH).C$ part of v's parent matches the $L.(IH).C$ part of u's label. We can deduce the $L.(IH).C$ part of v's parent as follows. If ph is non-empty, then the $L.(IH).C$ part of v's parent is $L.(ph).C$. Otherwise, if ph is empty, then the $L.(IH).C$ part is L. Note that it is impossible to deduce the PH component of the parent.

A-D relation: Given two nodes u and v, if we want to verify whether v is u's ancestor node, we consider two cases: (1) If v's label is a prefix of u's label, then v is an ancestor of u. (2) If u and v have the same sequence of components except for the SIH and PH and the SIH of v is lexicographically larger than the SIH of u, then v is an ancestor of u.

Sibling Relation: To determine if two nodes u and v are siblings, we check if they have the same parent. If we want to detect preceding/following sibling relation, we consider two cases: (1) If both u and v do not have a PH in their labels, we just compare their last component in lexicographic order. (2) If either u or v has a PH component, then the node (say v) with a PH component has one or more nodes inserted above it. If there is LS component in u's label, u precedes v. Otherwise, u follows v. The node u precedes v if the LS component of u is less than the LS component in v when both of the nodes have LS.

D. Compressed 2-D VLEI code

In DO-VLEI Code, ".1", "0" and "1" are compressed into (10), (0) and (11) respectively. In the 2-D VLEI Code, there are PH, IH, LS and normal VLEI code components, so we need 2 bits to separate 4 types of components. We use (01) to represent PH, use (10) to represent IH, use (11) to represent LS and use (00) to represent normal VLEI code. The question is where we should put the two bits that will not cause conflict in the future decoding. We put these two bits after the (10) which represents ".1" of DO-VLEI code. If we want to decide the type of component when decoding, we should scan two bits after we determine the position of "." and decide the type of component. Algorithm 1 shows the details of compressing 2-D VLEI Code. For example, 1.100.101.110.1 is compressed into 11100000100001110001101000. Because all of the DO-VLEI codes begin with 1 and are normal VLEI Code, the first component does not use additional two bits to represent the type of component. $1.(11).100.11.[1]$ is compressed into 111010111000001011111001. The first two bits of each component are 10(except the first component), and the two bits following represent the type of component. Therefore, the compressed 2-D VLEI code can be uniquely decoded into the original 2-D VLEI codes.

Algorithm 1 The Algorithm of compressing 2-D VLEI Code.

Input:
The label of 2-D VLEI Code before Compressing, $vlei$;
Output:
The Compression of 2-D VLEI Code, $cvlei$;

```
for i = 0 to i < vlei.length() do
  if vlei.charAt(i) ==' 1' then
    cvlei = cvlei + "11"
  else {vlei.charAt(i) ==' 0' }
    cvlei = cvlei + "0"
  else {label.charAt(i) ==' .'&&label.charAt(i+1) ==' (' }
    cvlei = cvlei + "10" + "10"
    i = i + 2
  else {label.charAt(i) ==' .'&&label.charAt(i+1) ==' [' }
    cvlei = cvlei + "10" + "01"
    i = i + 2
  else {label.charAt(p) ==' .'&&label.charAt(i+1) ==' {'}
    cvlei = cvlei + "10" + "11"
    i = i + 2
  else {label.charAt(p) ==' .'}
    cvlei = cvlei + "10" + "00"
    i = i + 1
  end if
end for
return cvlei;
```

TABLE I.
EXPERIMENTAL ENVIRONMENT

CPU	$Pentium(R)DualCoreE5300(2.60GHz)$
Memory	DDR2 2048MB
OS	Windows XP Professional
Memory	DDR2 2048MB
Java	1.6.0_02
DB	Office Access 2003

IV. STRUCTURAL INFORMATION EXTRACTION USING 2-D VLEI CODE

In this section, we will discuss the method of extracting node's information from 2-D VLEI Code. Structural information includes depth information, the label and name of parent node, the label and name of any ancestor nodes. We can also decide the inter-node relationships (P-C relation, A-D relation and sibling relation) between two nodes according to their labels.

Because the 2-D VLEI Code is also prefix-based labeling scheme, it is important to find the locations and count the number of the delimiters in 2-D VLEI Code. If we get the number of the delimiters, we will know the depth of node, and we only need to extract the prefix code before the rightmost delimiter to get the label of parent node. We will get node's name after querying the table that storing the labels and names of all the nodes for each XML document. To get any level ancestor nodes information, we repeat the method of parent node information extraction. In Compression of 2-D VLEI code part of Section III, we have described the method of finding the location of the delimiters and decoding labels. If given two nodes' labels, we should decode labels fist and then decide their inter-node relationships according to the method described in Structural Information of Section III.

V. EXPERIMENTAL EVALUATION

Experimental environment is shown in Table 1. Because the 2-D VLEI Code and OrdPathX are both support internal node insertions, we perform experiment to compare their performance, such as the speed of producing the node's label, the storage consumption and the efficiency of extracting information from the label. The XML documents used for the experiments were sourced from the XML Data Repository [10], we select 13 XML documents, and Table 2 shows the details of these XML documents, including the document name, document size, and the number of elements, and the maximum depth. The XML document sizes range from 1KB to 1.7MB.

A. Efficiency of Producing Labels

First, to compare the 2-D VLEI code with OrdPathX in the efficiency of producing labels, we designed experiment to calculate the label of 2-D VLEI code and OrdPathX of all the nodes in XML document, at the same time record the time of computing labels of all nodes of a XML document. Figure 7 shows the label generation time ratio for computing labels of all nodes of a XML document, using the 2-D VLEI code and OrdPathX. From this figure, we can see that the execution times for computing labels of all nodes using the 2-D VLEI code are about 15% less than the execution times using OrdPathX. The main reason is that compressing each component in OrdPathX needs to find a suitable record in the prefix schema.

TABLE II.
XML DOCUMENTS

document(.XML)	size(byte)	elements	maxdepth
region	787	21	3
nation	4,568	126	3
ubid	20,320	342	5
321gone	24,516	311	5
yahoo	25,421	342	5
supplier	29,250	801	3
ebay	35,562	156	5
reed	283,655	10546	4
SigmodRecord	478,416	11526	6
customer	515,660	13501	3
part	618,181	20001	3
wsu	1,647,864	74557	4
mondial-3.0	1,784,825	22423	5

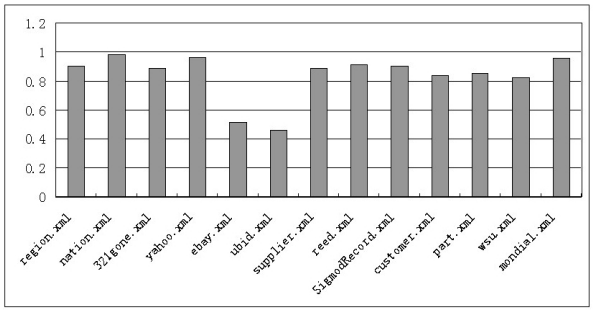


Figure 7. Label generation time ratio of 2-D VLEI code to OrdPathX.

B. Average Label Size

We then calculated and compared the average label size of each XML document labeled by the 2-D VLEI code and OrdPathX. Figure 8 shows the average label size ratio of the 2-D VLEI code to OrdPathX for each XML document and from which we can learn that the average label size using the 2-D VLEI code is about 15% on the average smaller than that using the OrdPathX. This is because that the compression of .1, 0 and 1 is as short as possible in 2-D VLEI code and then the total size of 2-D VLEI code will be the shortest. Moreover, in OrdPathX, skipping even numbers makes ORDPATH labels less compact, and the reserved space is not possible to be reduced, so this makes the compressed OrdPathX label longer. It means that the proposed labeling scheme also outperforms the OrdPathX in storage consumption.

C. Label Information Extraction

We also performed experiments to calculate the time of extracting information from the stored labels of each document. The information includes: 1) node's information (the depth, label after decoding, the node's name); 2) information of the parent node (the depth, label after

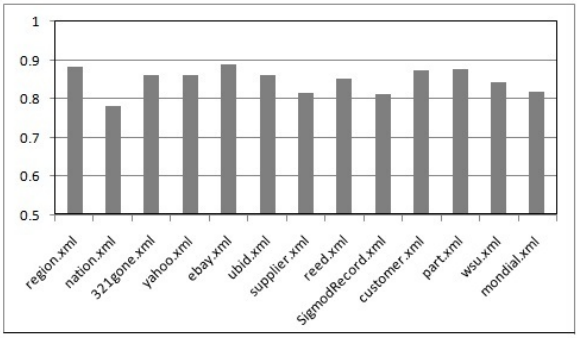


Figure 8. Label size ratio of 2-D VLEI code to OrdPathX.

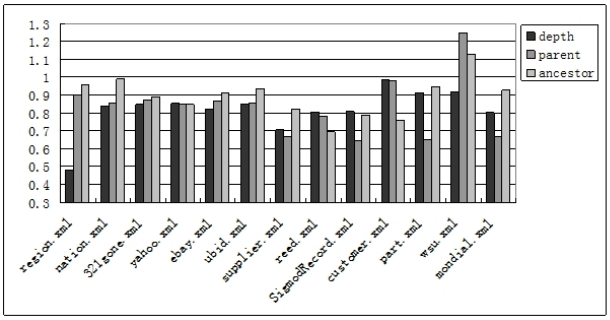


Figure 9. Execution time ratio of 2-D VLEI code to OrdPathX.

decoding, the node's name); 3) information of the ancestor nodes (the depth, label after decoding, the node's name). Figure 9 shows the execution time ratio for extracting self information, information of the parent node, and information of the ancestor nodes using the 2-D VLEI code and OrdPathX. From the experimental results we can see 2-D VLEI code can achieve high performance in structural information extraction. This is because the delimiter detection in OrdPathX requires traversal from the head through the whole code and refers to the prefix schema for determining each delimiter. While using the 2-D VLEI Code does not need to refer to any prefix schema, just counting the number of consecutive "1".

D. Efficiency of Inserting Labels

In the experiment, we performed parent insertions. We selected 30% of the total nodes in each XML documents and insert one parent for each selected node. Figure 10 shows the total insertion time ratio of 2-D VLEI code to OrdPathX, from which we can learn that the insertion time using 2-D VLEI code is about 47% on the average smaller than that using the OrdPathX, which means that the proposed method also outperforms the OrdPathX method in internal node insertion.

VI. Conclusion

Inspired by the method of inserting internal nodes of OrdPathX, in this paper we proposed 2-D VLEI code based on the DO-VLEI code. The 2-D VLEI code supports internal node insertions and does not need any prefix schema when compressing and decoding. We performed

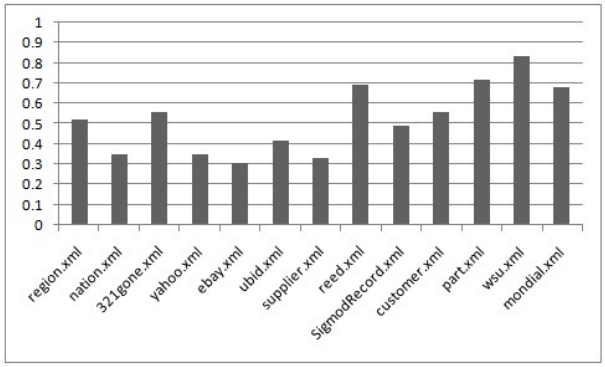


Figure 10. Insertion time ratio of 2-D VLEI code to OrdPathX.

experiments to evaluate the efficiency of producing labels, the storage consumption and the querying performance of 2-D VLEI code we proposed, and compared those with the OrdPathX. And we also compared the 2-D VLEI code with OrdPathX in internal node insertion. Our experimental results indicate that the 2-D VLEI code outperforms the OrdPathX in the above-mentioned four aspects.

Acknowledgment

This work was partially supported by SRF for ROCS, SEM, Doctoral Fund of Ministry of Education of China, the Fundamental Research Funds (DUT10JR02) for the Central Universities, China.

References

[1] Jing Cai and Chung Keung Poon. OrdPathX: Supporting Two Dimensions of Node Insertion in XML Data. In *Proceedings of DEXA*, pages 332–339, 2009.

[2] Paul Frederick Dietz. Maintaining Order in a Linked List. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 122–127, 1982.

[3] Su Cheng Haw and Chien Sing Lee. Node Labeling Schemes in XML Query Optimization: A Survey and Trends. *IETE TECHNICAL REVIEW*, 26(2):88–100, 2009.

[4] Kazuhito Kobayashi, Wenxin Liang, and Dai Kobayashi. VLEI code: An Efficient Labeling Method for Handling XML Documents in an RDB. In *Proceedings of ICDE*, pages 386–387, 2005.

[5] Kazuhito Kobayashi and Haruo Yokota. Evaluation of XML Laveling Methodss using Endless Insertable Code VLEI. In *DEWS2004*, 2004.

[6] Quanzhong Li and Bongki Moon. Indexing and Querying XML Data for Regular Path Expressions. In *Proceedings of the VLDB*, pages 361–370, 2001.

[7] Wenxin Liang, Akihiro Takahashi, and Haruo Yokota. A Low-Storage-Consumption XML Labeling Method for Efficient Structural Information Extraction. In *Proceedings of DEXA*, pages 7–22, 2009.

[8] Patrick O'Neil, Elizabeth O'Neil, and Shankar Pal. ORDPATHs:Insert-Friendly XML Node Labels. In *Proceedings of ACM SIGMOD Conference*, pages 903–908, 2004.

[9] Igor Tatarinov, Stratis D. Viglas, Kevin Beyer, Jayavel Shanmugasundaram, Eugene Shekita, and Chun Zhang. Storing and Querying Ordered XML Using a Relational Database System. In *Proceedings of ACM SIGMOD Conference*, pages 204–215, June 2002.

[10] Xml data repository. http://www.cs.washington.edu/research/xmldatasets/.
[11] Liang Xu, Zhifeng Bao, and Tok Wang Ling. A Dynamic Labeling Scheme Using Vectors. In *Proceedings of DEXA*, pages 130–140, 2007.
[12] Liang Xu, Tok Wang Ling, and Huayu Wu. Labeling dynamic xml documents:an order-centric approach. *IEEE Transactions on Knowledge and DataEngineering*, 2010.

Jie Chen received her B. Sc. in Software Engineering from School of Software, Dalian University of Technology in 2010. She is currently a master student with Dalian University of Technology. Her main research interests include XML databases and XML labeling.

Wenxin Liang received his B.E. and M.E. degrees from Xi'an Jiaotong University, China in 1998 and 2001, respectively. He received the Ph.D. degree in Computer Science from Tokyo Institute of Technology in 2006. He was a Postdoc Research Fellow, CREST of Japan Science and Technology Agency (JST) and a Guest Research Associate, GSIC of Tokyo Institute of Technology from Oct. 2006 to Mar. 2009. His main research interests include XML Data Processing and Management, XML Storage, Indexing, Labeling and Querying Techniques, XML Keyword Search, Web-based IR, Knowledge Discovery and Management, etc. He is currently an associate professor at School of Software, Dalian University of Technology, China. He is a senior member of China Computer Federation (CCF), and a member of IEEE, ACM, ACM SIGMOD Japan Chapter and Database Society of Japan (DBSJ).

Haruo Yokota received the B.E., M.E., and Dr. Eng. degrees from Tokyo Institute of Technology in 1980, 1982, and 1991, respectively. He joined Fujitsu Ltd. in 1982, and was a researcher at ICOT for the Japanese 5th Generation Computer Project from 1982 to 1986, and at Fujitsu Laboratories Ltd. from 1986 to 1992. From 1992 to 1998, he was an Associate Professor in Japan Advanced Institute of Science and Technology (JAIST). He is currently a Professor at Department of Computer Science in Tokyo Institute of Technology. His research interests include general research area of data engineering, information storage systems, and dependable computing. He is an associate editor of the VLDB Journal, a chair of ACM SIGMOD Japan Chapter, a trustee member of IPSJ and the Database Society of Japan (DBSJ), a fellow of IEICE and IPSJ, and a member of JSAI, IEEE, IEEE-CS, ACM and ACM-SIGMOD.

A Hop-by-hop Cross-layer Congestion Control Scheme for Wireless Sensor Networks

Guowei Wu, Feng Xia*, Lin Yao, Yan Zhang, and Yanwei Zhu
School of Software, Dalian University of Technology, Dalian 116620, China
Email: {wgwdut, fxia, yaolin}@dlut.edu.cn

***Abstract*—Congestions in wireless sensor networks (WSNs) could potentially cause packet loss, throughput impairment and energy waste. To address this issue, a hop-by-hop cross-layer congestion control scheme (HCCC) built on contention-based MAC protocol is proposed in this paper. According to MAC-layer channel information including buffer occupancy ratio and congestion degree of local node, HCCC dynamically adjusts channel access priority in MAC layer and data transmission rate of the node to tackle the problem of congestion. Simulations have been conducted to compare HCCC against closely-related existing schemes. The results show that HCCC exhibits considerable superiority in terms of packets loss ratio, throughput and energy efficiency.**

***Index Terms*— Wireless sensor networks, congestion control, cross-layer, hop-by-hop, priority**

I. INTRODUCTION

Wireless sensor networks (WSNs) have emerged as an innovative technology that can be applied in a wide range of areas including e.g. environment monitoring, smart spaces, medical systems, and robotic exploration [1]. It has been one of the hot research topics in recent years. The general task of a WSN is to perceive, collect and process information in a cooperative way in the region covered by sensor nodes, and to deliver the information to destination node via certain communication paths. In a sensor node, as data traffic becomes heavier, packets might be put into the node's buffer and have to wait for access to the medium that is shared by a number of communication entities. In such situations, congestion happens in the network. If network congestion becomes severe, certain packets will be dropped due to limited buffer size. This will potentially result in loss of packets, decrease in throughput, and waste of energy. For these reasons, congestion control is a critical challenge facing WSNs [2, 3].

In most cases, conventional congestion control schemes simply reduce the transmission rate at transport layer to relieve network congestion. As a consequence, they cannot maintain stable network throughput. Moreover, existing congestion control algorithms usually do not take into consideration the impact of energy efficiency. The extra signal transmission for the purpose of congestion control and the retransmission of dropped data packets will cause significantly large energy consumption, thus reducing the network life cycle. In addition, in many-to-one multi-hop routing, most of the existing algorithms may cause packets originating from sensors close to the congestion node to have a higher probability of being dropped, which is generally known as the unfairness problem.

To address these shortcomings, a hop-by-hop cross-layer congestion control scheme, namely HCCC, is proposed in this paper. HCCC shares the MAC layer channel information with transport layer and controls network congestion by adjusting data transmission rate and channel access priority. Simulations have been conducted to evaluate the performance of the proposed approach against other related solutions. The results are presented and analyzed.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the HCCC algorithm and gives theoretical analysis of its feasibility. Section 4 presents simulation results and performance analysis. Section 5 concludes the paper.

II. RELATED WORK

A number of congestion control protocols have been proposed for WSNs. The end-to-end and hop-by hop congestion control are two general methods for traffic control in WSNs. The end-to-end control can impose exact rate adjustment at each source node and simplify the design at intermediate nodes. However, it results in slow response and depends highly on the round-trip time (RTT). In contrast, hop-by-hop congestion control has faster response.

CODA (Congestion Detection and Avoidance) [4] is a typical congestion control mechanism in WSNs. It contains three basic strategies: congestion detection based on receiving, open-loop hop-by-hop feedback and multiple source rate adjustment in closed loop. CODA guarantees that throughput satisfies the accurate request by adjusting rate in closed loop way. However, it may cause seriously source rate shaking because of the AIMD (Additive Increase Multiplicative Decrease) strategy, and sensor nodes may deplete extra energy by monitoring channels periodically. ESRT (Event-Sink Reliable Transport) protocol [5] mainly guarantees reliable transmissions and controls congestion by changing and transforming the network state. The SenTCP [6] protocol

* Corresponding Author: Feng Xia (f.xia@ieee.org)

doi:10.4304/jsw.6.12.2434-2440

uses a more accurate method to detect congestion than CODA. It is good for adjusting the sensor node rates properly.

Cross-layer design can share the information of wireless medium in MAC layer and physical layer with up layers (i.e. network layer, transport layer, and application layer), which can allocate the network resource effectively and hence improve the network performance. Some cross-layer congestion control algorithms have been proposed in recent years. For instance, Hull *et al.* [2] proposed a cross-layer congestion control scheme named Fusion. It exploits three techniques to achieve cross-layer processing: hop-by-hop flow control, rate limiting source traffic when transit traffic is present, and a prioritized MAC protocol. The PCCP [7] algorithm assigns different priorities to every node. It uses cross-layer optimization approach to detect congestion degree and mitigate congestion, where nodes' rates and flows are adjusted according to the priorities. Lin and Shroff [8] presented a cross-layer optimization scheme for multi-hop wireless network, which focuses on how the performance of congestion control will be impacted by imperfect scheduling algorithms. Chiang [9] proposed to jointly optimize congestion control and power control in cross-layer manner. The ANAR [10] mechanism is another cross-layer optimization scheme, which combines transport-layer congestion control and network-layer routing protocol. The Cross-Layer Active Predictive Congestion Control (CL-APCC) scheme [11] for improving the performance of networks applies queuing theory to analyze data flows of a single-node according to its memory status, combined with the analysis of the average occupied memory size of local networks. In order to ensure the fairness and timeliness of the network, the IEEE 802.11 protocol is revised based on waiting time, the number of the node's neighbors and the original priority of data packets. The sending priority of the node is adjusted dynamically. DiffQ [12] provides practical adaptation and implementation of differential backlog that involves a cross-layer optimization of both congestion control and MAC scheduling in real multi-hop wireless networks. ACT (Adaptive Compression-based congestion control Technique) [13] is an adaptive compression scheme for packet reduction in case of congestion. The main problem of ACT is its high complexity. In addition, there are a number of research works attempting to increase the sensor node data transmission throughput, packet delivery ratio and data security via multipath routing [14-18].

Although the existing schemes [3, 19-24] play important roles in improving performance of WSNs, designing an effective congestion control scheme is still a challenging issue in WSNs. In this paper, a hop-by-hop cross-layer congestion control scheme is introduced. The major differences between this work and the aforementioned approaches include the following aspects:

(1) HCCC shares the MAC layer channel information with transport layer, which is used to adjust local channel access probability. When congestion occurs, congestion information can be quickly fed back to upstream nodes, while local congestion can also be alleviated as soon as possible and the local node's buffer queue can avoid being overflowed.

(2) In order to relieve congestion and keep stable network throughput, our HCCC algorithm dynamically adjusts the channel access priority to multiplicatively decrease or linearly increase data transmission rate.

III. Hop-by-hop Cross-layer Congestion Control

The HCCC algorithm can be built on various contention-based MAC protocols that are widely used in wireless sensor networks. In this work, we adopt the S-MAC protocol [25]. HCCC is composed of three main parts: (1) congestion detection, (2) feedback signal sending and local congestion processing, and (3) feedback signal processing in upstream node. The whole process of the HCCC algorithm is shown in Fig. 1. Suppose that node A, node B and node C are arbitrary intermediate nodes in a WSN, all of which perform the congestion control algorithm. All of them compute their own local congestion information. Node C feeds back its congestion information to its upstream node B. After receiving the feedback signal, node B will add its own congestion information into the feedback signal from node C, and then relay the new feedback signal to its upstream node A. Node B also carries out local congestion processing and feedback signal processing to relieve the congestion within the downstream node C and itself. Node A processes the feedback signal in the same manner with node B, and the feedback signal will be sent to the source node hop-by-hop. Finally the source node will adjust its data transmission rate to relieve congestion. In the following subsections, we will describe the HCCC scheme in detail through elaborating on the used congestion detection method, feedback signal generation and transmission sending method, feedback signal and local congestion processing method.

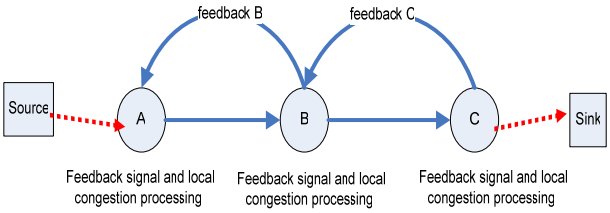


Figure 1. Illustration of proposed scheme

It is worth noting that frequent hop-by-hop transmission of feedback signals would consume significant amount of the node's energy, which is not conducive to prolong network lifetime. To avoid this problem, the HCCC algorithm adopts implicit notice mechanism. The feedback signal is attached in the RTS/CTS control frame of MAC protocol. The MAC layer manages the radio channel and sends the congestion signal to the source node hop-by-hop, which can avoid energy waste caused by broadcast.

A. Congestion Detection

In order to satisfy the accuracy and low-cost requirements of network congestion detection, HCCC adopts the similar detection mechanism as SenTCP [6] does. We define two parameters: congestion degree C_d and buffer occupancy ratio B_r. Congestion degree indicates the changing tendency of buffer queue.

The value of C_d is defined as follows.

$$C_d = T_s / T_a \quad (1)$$

$$T_a = (1-p) \times T_s + p \times (t - t') \quad (2)$$

$$T_s = (1-p) \times (t - t^{'}) + p \times t_s \quad (3)$$

In the above equation, T_a is the interval between the arrival of two adjacent data packets in MAC layer, t is the arrival time of the data packet, t' is the arrival time of last data packet, T_s is the average processing time of data packets in local node, t_s is the transmission time of data packets, and p is an adjustable parameter, which is set to 0.3 in this work. The value of T_s is updated when a data packet is sent out. If $C_d > 1$, the arrival rate is bigger than the departure rate of data packets, indicating that congestion may possibly happen in the near future. Let the threshold of buffer occupancy ratio be B_{max}. If $B_r > B_{max}$, HCCC can judge that congestion happened. To lower cost, HCCC adopts the most direct way to detect congestion, i.e., using the buffer queue length in local node to detect congestion. When the number of data packets in buffer queue exceeds the threshold value, it is believed that the data packets will overflow the buffer queue in short time, and congestion will happen in the local node. Given below is the congestion detection algorithm.

Algorithm 1: Congestion Detection

Input: Node's buffer occupancy ratio B_r and congestion degree C_d
Result: Node state change
1: Initialize node information;
2: Compute B_r and C_d;
3: if $C_d > 1$ && $B_r > B_{max}$ then
4: Set the node state to congestion state and perform local congestion processing mechanism;
5: Send congestion information (i.e. feedback signal) to upstream node and examine the feedback signal from downstream node;
6: end if
7: if $C_d > 1$ && $B_r < B_{max}$ then
8: Adjust local data transmission rate;
9: end if
10: if $C_d < 1$ && $B_r < B_{max}$ then
11: Set local node state to non-congestion and send node state information to upstream node;
12: end if

As we can see, our congestion detection scheme can not only enable the upstream node to quickly decrease data transmission rate after receiving the congestion signal, but also adaptively adjust local data transmission rate according to congestion tendency.

B. Feedback Signal Generation and Transmission

The second phase of HCCC is to generate feedback signal for upstream node and process local congestion. There are three issues that need to be solved in this phase: (1) when to transmit the feedback signal? (2) how to transmit the feedback signal? (3) how to process the congestion locally?

In general, a sensor node may have three states: transmitting, receiving and sleeping. Since a sensor node can only execute the congestion control algorithm when it is not in sleeping state, there are two mechanisms to generate congestion feedback signal: one is to generate feedback signal before the node transmits data packets, and the other is to generate feedback signal before the node receives data packets. Fig.2 and Fig.3 illustrate these two mechanisms respectively, in which node B is the local node, node A is upstream node, and SIFS is the shortest time that physical hardware requires to transform from receiving or detecting state to sending state. The mechanisms are described in more detail in the following.

1) With the first mechanism (Fig. 2), when the local node is ready to transmit data packets, HCCC performs local congestion detection, and adjusts channel access priority according to the congestion condition. The node then sends the congestion information attached in RTS (Request to Send) packet to the upstream node. The upstream node adjusts its channel access priority according to the received congestion information when it begins to transmit a new data packet in next time slot.

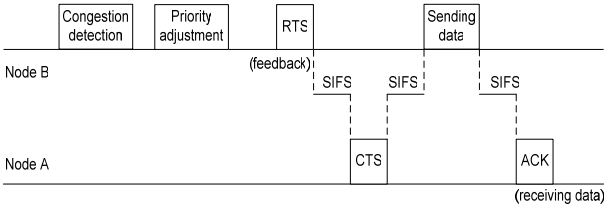


Figure 2. Feedback Signal Generation Method 1

2) With the second mechanism (see Fig. 3), the node performs local congestion detection after receiving the RTS request from the upstream node, then replies to the upstream node with the congestion signal attached in CTS (Clear To Send) packet. The upstream node adjusts channel access priority after receiving the congestion signal when the next data packet is transmitted. The local node adjusts local channel access priority according to the congestion condition when a data packet needs to be sent out.

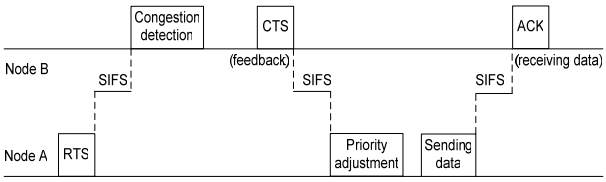


Figure 3. Feedback Signal Generation Method 2

In this work we use the first mechanism for generating and transmitting feedback signals. The reason behind is

that it is better to detect congestion and send congestion information before data packets are sent out.

C. Feedback Signal and Local Congestion Processing

The rate adjusting strategy could affect network communication performance significantly, especially network throughput and transmission fairness. When the upstream node processes feedback signals from the downstream node, it should also consider its own congestion condition.

Assuming that the upstream node receives congestion signal successfully, the feedback signal and local congestion processing method used in HCCC is shown in Algorithm 2, where W is the size of local channel competition window, W_{max} is the maximum size of channel contention window in the MAC protocol, and W_{min} is the minimum size of channel contention window. For example, in S-MAC, the range of channel contention window is [1, 63], while it is [15, 1023] in IEEE 802.11. We adopt the same window size as S-MAC. R is local node's data transmission rate. $\Delta R = 0.5 \times (R_{max} - R)$, where R_{max} is the maximum data transmission rate of local node during the period of time from the moment it responds to congestion feedback signal to present. Let B'_r be the buffer occupancy ratio of downstream node, which can be obtained from the feedback signal.

Algorithm 2: Feedback Signal and Local Congestion Processing

Input: Local buffer occupancy ratio B_r, feedback signal B'_r from downstream node
Result: Local channel contention window W, data transmission rate R
1: Initialize node information; $W = W_{max}$;
2: if $B'_r > B_{max}$ && $B_r > B_{max}$ then
3: $R \leftarrow 0.25 \times R$; $W = 0.5 \times (5 \times W \times B_r + 0.1 \times W \times 1/B'_r)$;
4: end if
5: if $B'_r > B_{max}$ && $B_r \leq B_{max}$ then
6: $R \leftarrow 0.5 \times R$; $W = 5 \times W \times B'_r$;
7: end if
8: if $B'_r \leq B_{max}$ && $B_r > B_{max}$ then
9: $R \leftarrow \min[0.5 \times R, R + \Delta R]$; $W = \min[10 \times W \times B'_r, 0.1 \times W \times 1/B_r]$;
10: end if
11: if $B'_r \leq B_{max}$ && $B_r < B_{max}$ then
12: $R \leftarrow R + \Delta R$; $W = 10 \times W \times B'_r$;
13: end if

During the hop-by-hop relay process of feedback signals, HCCC gives priority to the handling of local congestion. If $B_r > B_{max}$, which indicates that congestion occurs locally, the node will first send its congestion feedback signal to its upstream node regardless of whether the congestion happens in its downstream node. When $B'_r > B_{max}$ and $B_r \leq B_{max}$, which indicate that congestion happens in the downstream node and not in local node, if the last feedback signal sent out is generated by the local node, then the node will relay the downstream node's feedback signal to its upstream node; otherwise it will not receive the feedback information. This mechanism takes into account both the downstream node's congestion information and the status of the local node. It also avoids frequently relaying congestion information of downstream nodes which would otherwise cause energy waste. When $B'_r \leq B_{max}$ and $B_r > B_{max}$, which indicate that congestion occurs in local node and not in downstream node, the node will only deal with the local congestion without relay. In case of congestion, the local node will reduce its data transmission rate and reduce channel contention window to increase channel access probability. When there is no local congestion, if $B'_r > B_{max}$, indicating that congestion occurs in the downstream node, it will increase channel contention window by $W = 5 \times W \times B'_r$ and reduce the data transmission rate as $R = 0.5 \times R$; otherwise it will linearly increase the data transmission rate as $R = R + \Delta R$.

It is clear that HCCC exploits the AIMD strategy for transmission rate adjustment. The major purpose of using such a strategy is to relieve local congestion as soon as possible while keeping stable network throughput.

D. Feasibility Analysis

Theorem I: A sensor node's data transmission rate R is proportional to its channel access priority P_r, i.e. $R \propto P_r$

Proof: Suppose that R^i_{in} is the total input traffic rate of node i, R is the packet transmission rate at the node i towards node $(i + 1)$, and R^i_f is the packet forwarding rate in the channel, which depends on the channel access priority P_r, with high channel access priority implying high probability to access channel and thus yielding high packet forwarding rate. If R^i_{in} is smaller than (or equal to) R^i_f, R will be equal to R^i_{in}. Otherwise, R will be approximately equal to R^i_f. When congestion happens, i.e. $R^i_{in} > R^i_f$, the node's data transmission rate R is approximately equal to R^i_f. Since R^i_f is proportional to channel access priority P_r, R is proportional to channel access priority P_r, i.e., $R \propto P_r$. As a consequence, it is possible to change the transmission rate by adjusting the channel access priority.

Theorem II: Channel access priority P_r is inversely proportional to contention window size W, i.e. $P_r \propto \frac{1}{W}$.

Proof: Suppose that S is the set of source nodes, L is the set of links, and c_l is the maximum number of packets that can be transmitted in the link in each time slot. Consider a wireless sensor network with L links, each with a fixed capacity of c_l, and S source nodes with transmission rate of R_s ($s \in S$). In [8] and [26] it has been obtained that $R_s \propto \frac{1}{D(t)}$, where $D(t)$ is total network delay, including node processing delay, queuing delay and transmission delay. Assume there are N hops from the source to the sink. The queuing delay is random at each hop. Let the value of queuing delay at hop n be $t_{cs,n}$, whose mean value will be determined by the contention window size W, and is denoted by t_{cs}. It is held that

$t_{cs} \propto W$. The transmission delay will be fixed if the packet length is fixed, which is denoted by t_{tx}. Accordingly the entire delay $D(t)$ over N hops is:

$$D(t) = \sum_{n=1}^{N} (t_{cs,n} + t_{tx}) \qquad (4)$$

Therefore, we can get $R_s \propto \frac{1}{D(t)} \propto \frac{1}{W}$, i.e., R_s is inversely proportional to contention window size W. Since Theorem I proves that R is proportional to channel access priority P_r, channel access priority P_r is inversely proportional to contention window size W, i.e., $P_r \propto \frac{1}{W}$. Consequently, it is feasible to reset the channel access priority through adjusting contention window size.

IV. Performance Evaluation

To assess the performance of the proposal scheme, we simulate four congestion control schemes including HCCC, CODA [4], ESRT [5] and Fusion [2] using the NS2 simulator. We analyze the performance of these four mechanisms in term of packet loss ratio, throughput, average source transmission rate (i.e. average transmission rate of source nodes) and energy efficiency.

The simulation parameters are set as follows. 100 sensor nodes (including source nodes and sink nodes) are randomly distributed in a square region of 100m×100m. The nodes' communication radius is 30m. The routing protocol used is DSR (Dynamic Source Routing). The network bandwidth is 2Mbps. The transmission rate is 1Mbps. The threshold of buffer occupancy ratio B_{max} is set to 0.4. The range of channel contention window size is [1, 63]. The initial energy of sensor nodes is 0.1J. The main energy consumption of nodes for delivering data packets is 10^{-4}J/packet. The total buffer size is 500 data packets. The size of every packet is 200 Bytes. The offered load is 5 packets/second (pps).

A. Packet Loss Ratio

Fig. 4 depicts packet loss ratios (with respect to time) associated with the four congestion control schemes. It can be seen that the packet loss ratio with HCCC is lower than that of CODA and ESRT most of the time. The superiority is especially clear during the beginning stage of system running. HCCC yields almost the same packet loss ratio with Fusion.

B. Network Throughput

Fig. 5 compares network throughput for these four schemes. The results show that the throughput with HCCC is much higher than the other three schemes, especially when the bit error rate becomes larger. In addition, there is less significant fluctuation when HCCC is used.

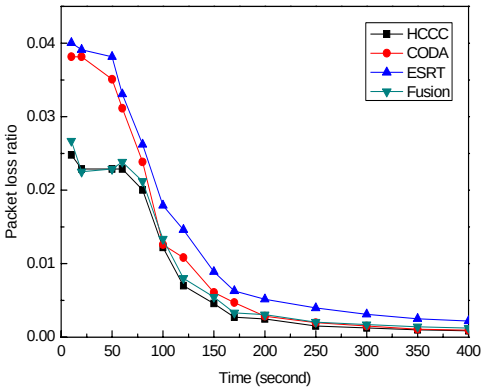


Figure 4. Packet loss ratio

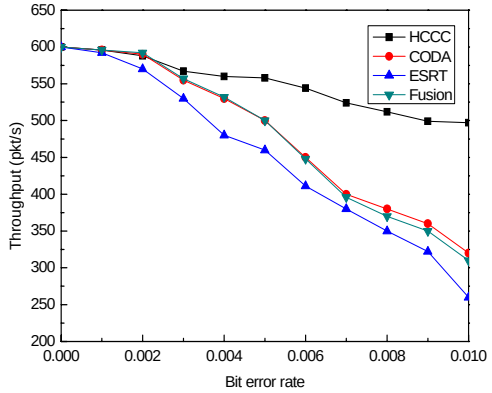


Figure 5. Network throughput

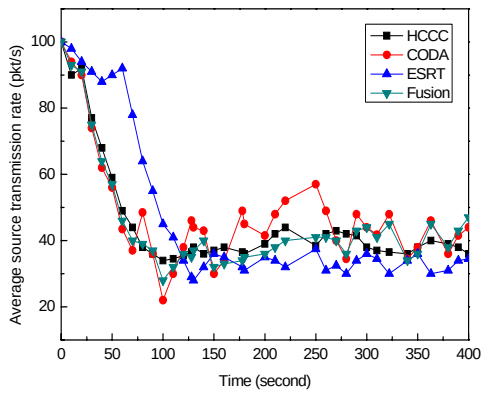


Figure 6. Average transmission rate of source nodes

C. Source Transmission Rate

Fig. 6 shows the average transmission rate of source nodes under different schemes. The source transmission rate with HCCC maintains in quite a stable level after the transient process. Though the performance of four schemes is comparable in this regard, CODA yields a bit more serious fluctuation in the steady state than the others.

D. Energy Efficiency

Fig. 7 gives the comparisons in terms of energy efficiency, which is calculated as the ratio of the sum of nodes' remaining energy to the sum of nodes' initial energy. We can see that HCCC has the highest energy efficiency, while ESRT performs worst in saving energy.

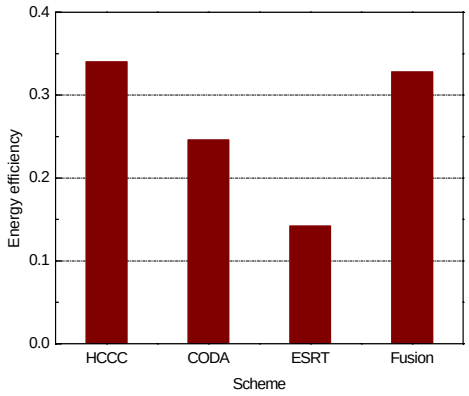


Figure 7. Energy efficiency

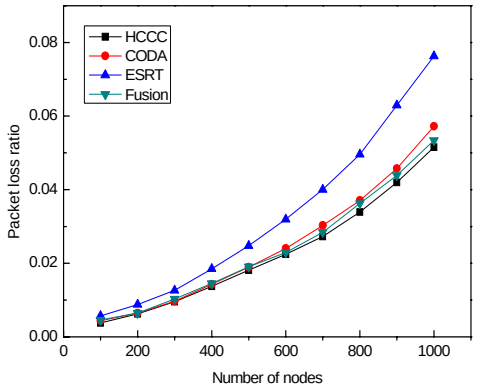


Figure 8. Packet loss ratio with respect to number of nodes

E. Discussions

In large scale WSNs, when congestion occurs in one node, the congestion will propagate to other nodes. Congestion regions would consequently form in the network. The more sensor nodes there are, the more packets will be dropped. To address congestion in networks in different scales, congestion control algorithms should have good adaptability. In Fig. 8, we compare the packet loss ratios with different schemes for different numbers of sensor nodes. It is clear that the packet loss ratio increases with the number of sensor nodes. Although HCCC can avoid the propagation of local congestion to its downstream node to some degree, its performance gets worse with the network scale increasing. Therefore, it is not suitable for large scale WSNs.

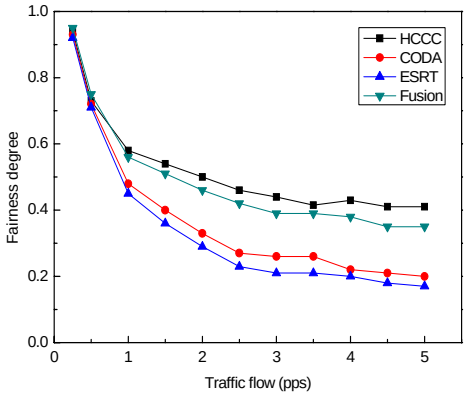


Figure 9. Fairness degree

In Fig. 9, we compare the fairness of the four schemes. In this set of simulations, the fairness degree [27], denoted Φ, is chosen as the metric, which can be computed as:

$$\Phi = \frac{\sum_{i=1}^{N} r_i}{N\sum_{i=1}^{N} r_i^2} \tag{5}$$

where N is the number of nodes and r_i is the average data transmission rate of node i. Obviously the value of Φ will vary with data traffic flow. As we can see from Fig. 9, the fairness under the four schemes becomes worse as the traffic flow increases. In particular, CODA and ESRT performs worse in terms of fairness than the others. Although the fairness issue has not been taken into account during the course of HCCC design, it can still yield get relatively better fairness performance than the other three schemes. However, it should be pointed out that there is much room for the improvement of the fairness of HCCC.

V. CONCLUSIONS

In this paper, a hop-by-hop cross-layer congestion control (HCCC) scheme has been presented. HCCC detects local congestion at proper moments, and delivers the congestion information to upstream nodes by exploiting the transmission of RTS and CTS frames. Meanwhile, it adapts the channel access priorities and data transmission rates of sensor nodes. Thus it can adaptively adjust the allocation of channel resource among sensor nodes. The presented simulation results demonstrate that our scheme has good performance in terms of packet loss ratio, throughput, source data transmission rate, and energy efficiency.

ACKNOWLEDGMENT

This work is partially supported by Natural Science Foundation of China under Grant No. 60903153, the Fundamental Research Funds for the Central Universities, and the SRF for ROCS, SEM.

REFERENCES

[1] C. Chong, S. Kumar and B. Hamilton, "Sensor networks: Evolution, opportunities and challenges," *Proceedings of the IEEE*, 9(18), pp.247-256, 2003.

[2] B. Hull, K. Jamieson and H. Balakrishnan, "Mitigating Congestion in Wireless Sensor Networks," *Proc. of the 2nd ACM Conf on Embedded Networked Sensor Systems*, pp.134-147, 2006.

[3] Wei-wei Fang, Ji-ming Chen, Lei Shu, Tian-shu Chu and De-pei Qian, "Congestion avoidance, detection and alleviation in wireless sensor networks," *J. Zhejiang Univ*, vol.11, no.1, pp.63-73, 2010.

[4] C. Wang, Y. Eisenman and A. Campbell, "CODA: Congestion Detection and Avoidance in Sensor Network," in *Proc. of ACM SenSys'03*, pp.1003-1016, 2003.

[5] Y. Sankarasubramaniam, O.B. Akan, I.F. Akyidiz, "ESRT: Event-to sink reliable transport in wireless sensor networks," in *Proc of the 4th ACM Symp on Mobile Ad Hoc Networking and Computing*, pp.177-188, 2003.

[6] Yueming Hu, Yueju Xue and Bo Li, "SenTCP: A hop-by-hop congestion control protocol for wireless sensor networks," *IEEE INFOCOM 2005*, pp.162-170, 2005.

[7] C. Wang, B. Li and K. Sohraby, "Upstream Congestion Control in Wireless Sensor Networks through Cross-layer Optimization," *IEEE Journal on Selected Areas in Communications*, vo.25, no.4, pp.786-795, 2007.

[8] Xiaojun Lin and Ness B.Shroff, "The Impact of Imperfect Scheduling on Cross-Layer Congestion Control in Wireless Networks," *IEEE/ACM Transaction on Networking*, vo.25, no.5, pp.302-315, 2006.

[9] Mung Chiang, "Balancing Transport and Physical Layers in Wireless Multi-hop Networks: Jointly Optimal Congestion Control and Power Control," *IEEE Journal on Selected Areas*, vol.23, no.1, pp.104-116, 2007.

[10] Yu-Pin Hsu and Kai-Ten Feng, "Cross-Layer Routing for Congestion Control in Wireless Sensor Networks," *Radio and Wireless Symposium*, pp.783-786, 2008.

[11] J. Wan, X. Xu, R. Feng and Y. Wu, "Cross-Layer Active Predictive Congestion Control Protocol for Wireless Sensor Networks," *Sensors 2009*, vol.9, no.10, pp.8278-8310, 2009.

[12] A. Warrier, S. Janakiraman, S. Ha and I. Rhee, "DiffQ: Practical Differential Backlog Congestion Control for Wireless Networks," *INFOCOM,* pp.19-25, 2009.

[13] Joa-Hyoung Lee and In-Bum Jung, "Adaptive-Compression Based Congestion Control Technique for Wireless Sensor Networks,"*Sensors 2010*, vol.10, no.4, pp.2919-2945, 2010.

[14] A. Razzaque and S. Hong, "Congestion Detection and Control Algorithms for Multipath Data Forwarding in Sensor Networks," in *Proc. of the 11th International Conference on Advanced Communication Technology (ICACT2009)*, pp. 651-653, 2009.

[15] M. Maimour, C. Pham and J. Amelot, "Load repartition for congestion control in multimedia wireless sensor networks with multipath routing," in *Proceedings of the 3rd International Symposium on Wireless Pervasive Computing (ISWPC '08)*, pp. 11-15, 2008.

[16] H. Yaghmaee and D. Adjeroh, "A New Priority Based Congestion Control Protocol for Wireless Multimedia Sensor Networks," in *International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pp. 1-8, 2008.

[17] A. Razzaque and S. Hong, "Congestion Detection and Control Algorithms for Multipath Data Forwarding in Sensor Networks," in *Proceedings of the 11th International Conference on Advanced Communication Technology (ICACT2009)*, pp. 651-653, 2009.

[18] K. Karenos, V. Kalogeraki and S. Krishnamurthy, "Cluster-based Congestion Control for Sensor Networks," *ACM Trans*. Sens. Netw. (TOSN), vol.4, no.1, pp.1-39, 2008.

[19] N. Sengottaiyan and R. Somasundaram, "A Modified Routing Algorithm for Reducing Congestion in Wireless Sensor Networks," *Eur. J. Sci. Res*, vol.35, no.4, pp.529-536, 2009.

[20] P. Reena and L. Jacob, "A Cross Layer Design for Congestion Control in UWB Based Wireless Sensor Networks," *Int. J. Sens. Netw,* vol.5, no.4, pp.223-235, 2009.

[21] M. Alam and S. Hong, "CRRT: Congestion-Aware and Rate-Controlled Reliable Transport in Wireless Sensor Networks," *IEICE Trans. Commun,* vol. E92-B, no.1, pp.184-189, 2009.

[22] K. Karenos, V. Kalogeraki and S. Krishnamurthy, "Cluster-based Congestion Control for Sensor Networks," *ACM Trans. Sens. Netw. (TOSN)*, vol.4, no.1, pp.1-39, 2008.

[23] Kai Lin, Lei Wang, Keqiu Li and Lei Shu, "Multi-Attribute Data Fusion for Energy Equilibrium Routing in Wireless Sensor Networks", *KSII Transactions on Internet and Information Systems*, vol.4, no.1, pp.5-24, 2010.

[24] Jian-Jun Lei and Gu-In Kwon, "Reliable Data Transmission Based on Erasure-resilient Code in Wireless Sensor Networks", *KSII Transactions on Internet and Information Systems*, vol.4, no.1, pp.62-77, 2010.

[25] Wei Ye, John Heidemann, and Deborah Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proceedings of the 21st IEEE INFOCOM,* pp.1567-1576, 2002.

[26] C. Wang, B. Li and K. Sohraby, "Upstream Congestion Control in Wireless Sensor Networks through Cross-layer Optimization," *IEEE Journal on Selected Areas in Communications*, vol.25, no.4, pp.786-795, 2007.

[27] B. Ruzena and E.C. Tien, "Congestion Control and Fairness for Many-to-One Routing in Sensor Networks," *ACM SENSYS*, pp.148-161, 2004.

Role Assorted Community Discovery for Weighted Networks

Ruixin Ma
School of software, Dalian University of Technology, Dalian, China
Email: teacher_mrx@126.com

Guishi Deng and Xiao Wang
School of Management Science and Engineering, Dalian University of Technology, Dalian, China
Email: Denggs@dlut.edu.cn, kara0807@126.com

Abstract—**This paper considers the difficulties in community discovery, and comes up with a community discovery algorithm on the basis of role assorted thoughts. Previous work indicates that a robust approach to community detection is the maximization of inner communication and the minimization of the in-out interaction. Here we show that this problem can be solved accords to the role assorted method which give distinguish labels to vertices in the same community. This method leads us to a number of possible algorithms for detecting community structures in both unweighted and weighted networks. The applicability and expandability of algorithms proposed are illustrated with application to a variety of computer-generated networks and real-world complex networks.**

Index Terms—**community discovery, role assorted thoughts, robust approach, distinguish label**

I. INTRODUCTION

Networks have attracted considerable recent attention in biology and other fields as foundation for the mathematical presentation of a variety of complex systems. [1-4]. It seems that all complex systems can be abstracted as networks which are combined by all kinds of interacting units. Therefore, network provides a totally novel, intuitional method for complex system research. It is of great value to find community structures in complex network. E.g. communities in social network are used to reveal the groups of users have similar interests, habits and background[5-7]. Community discovery in citation system helps readers to quickly find the papers they want. Communities in biochemical networks help researchers to find the functional related units[8-11], to name but a few.

The research of community discovery is first proposed by Girvan and Newman[12,13]. They put forward some classic CDAs such as G-N and fast G-N, besides, Newman also proposed a method to measure the results of CDAs which was called modularity. As we all know, modularity is indeed a good way to test whether the discovery results is good or not, however, it is limited in unweighted networks while not suit for weighted networks. Especially now that the development of SNS[14] has become an inevitable tendency all over the world, CDA should try its best to work for good social network services. In that case, some scholars are not content with singly find community structures in the entire network, but prefer to explore the structure inside a community and refine the community division results.

Advertising practitioner Fang Shouxing said that there is a "law of special" in the procedure of website promotion. That is to say, there are three kinds of people play very important roles during the course of information dissemination. First, authorities with great stores of knowledge in one or more frontiers. For websites and application platforms, lots of successful founders on the internet are either authorities or the contact of authorities, this kind of people have special perspectives to convene experts. In other words, they are the leaders in their communities. Second, liaisons, this kind of people have talents in social interactions. Generally speaking, liaisons keep in touch with different communities at the same time. Six degrees of separation tells us that every two people in the world can reach each other within six degrees. However, it is not means any one in the world is able to reach the others but only some special ones who have access to different communities can. Most people keep in touch with world by these few special ones. Third, recommender, they take charge of "the last mile" which means they persuade people to buy the things they recommend. Most recommenders are just ordinary users, they may far away from the authoritative experts and the smooth liaisons, but they can persuade our target user to accept their idea. Whether information can be propagated like virus or not depends on how many powerful recommenders are working for you. Make every user be your recommender maybe every website's dream. Recommenders on SNS may be the target user's most trusted people, or their nearest neighbors.

The above mentioned three kinds of people are often called the opinion leaders on the website. Most users take leaders' opinions as their references to browse pages or to

doi:10.4304/jsw.6.12.2441-2448

buy goods while they do not sure what they want from the Internet.

As far as our goals in this paper are concerned, a very useful method is that taken by social network analysis with the set of knowledge known as collaborative filtering[15-19] and the theory of space vector model[20]. This knowledge is aimed at mining natural structures in social networks, based on both the adjacent relationship and the strength of connection between vertices which is called role assorted community discovery algorithm. RACDA is a kind of agglomerative algorithm who focuses on the addition of vertices to different communities. In this agglomerative method, similarities are calculated by one method or another between vertices and communities.

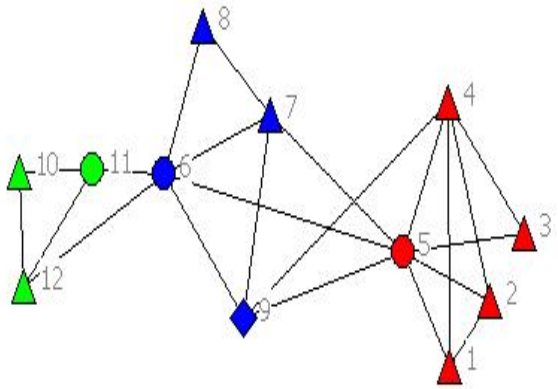


Figure 1. A small network with community structure of the type considered in this paper.

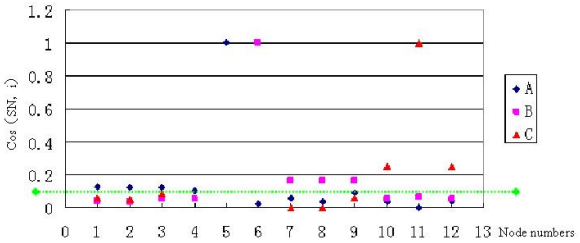


Figure 2. The diagram illustrates the type of output generated by the our algorithms.

Frankly speaking, agglomerative methods have some problems on the procedure of finding the right community structure. One concern is that they fail with some frequency to find the correct communities in networks where the community structure is known, which makes it difficult to place much trust in them in other cases. Another is their tendency to find only the cores of communities and leave out the periphery[21]. Besides, in cases those vertices only have a single link to a specific community, agglomerative methods often fail to place such vertices correctly.

However, in this paper, we come up with a novel agglomerative CDA which based on the theory of space vector model. This method has been relatively little studied in the previous literature, either in social network theory or elsewhere. Here we introduce the concept of community seed, community eigenvector and liaisons. In this agglomerative method, we start with finding the community seed and use community eigenvectors to represent the adjacent circumstances in each community, take figure 1 as example; community seeds 5, 6 and 11 are primarily found. By calculating the similarity between free vertices and communities repeatedly, we get the entire network's natural structures and the illustration diagram for the communities we get as figure 2 shows.

The approach we take follows roughly these lines, but adopts a some what different heuristic viewpoint. Rather than starting with randomly choosing vertices in the network, we make a list of ranking nodes in decreasing order of fitness, which makes sure the priority of vertices with better position. We pay much more attention to the similarity between vertices and communities instead of similarities in node pairs. Besides, by doing some marks to the vertices which are divided into the same communities, we can also find the liaisons among different communities. How this idea works out in practice will become clear in the course of the presentation.

Briefly then, the outline of this paper is as follows. In section II we describe the crucial concepts behind our algorithm and show how these concepts works in the implementation of our method. In section III, we describe the detailed implementation of our algorithm. In section IV, we give a number of applications of our algorithms to particular networks. At the end of this paper, we give a conclusion.

II. COMMUNITY DETECTING FOR SOCIAL NETWORK SERVICE

In this paper, we present a new algorithm for network clustering. Our discussion focuses primarily on networks with a type of weighted edges while generalization to less complicated network types is certainly possible.

There are three central features that distinguish our algorithms from those that have preceded before. First, our algorithms are aimed at finding three kinds of special people on the network and use the influence of the special to find community structures behind the network. Second, we introduce the application of space vector model in community discovery. Third, characteristics of community are represented by community eigenvectors which is totally different from the metric's eigenvector.

To make things more concrete, we use example to illustrate how the space vector model works in this algorithm.

Both the adjacent relation and the intimacy relation are c represented by vectors. Before further explanation, we firstly define that the adjacent matrix of network is *Matrix* and the intimacy relation matrix of network is *R-Matrix*. Each row in adjacent matrix shows a vertex's adjacent relation with others while each row in *R-Matrix* shows the intimacy degree of each pair of vertices, which is showed as $V_i =< e_{i,1}, e_{i,2} \ldots e_{i,n} >$. The intimacy relation vector of communities in weighted networks can be

calculated as formula (1). Values on the edge are added to the community eigenvector to show the intimacy between vertex and the existing community. Both the adjacent topology and the intimacy between vertices become the factors to judge the adscription of vertices.

foreach vertex i in SN

$$V_{SN} = V_{SN} + V_i < e_{i,1}, e_{i,2}, \ldots e_{i,n} > \quad (1)$$

The totting-up of relation during the procedure of structure division makes the close-knit vertices come closer while expand the gaps between different communities.

This measure is only a suggestion; many others are possible and may well be appropriate for specific applications.

Another way in which our method differ from previous ones is in the procedure of doing distinguish marks for vertices with different characteristics, which means network managers are able to provide different levels of protection for vertices in the same community. We put vertices into three classes in the procedure of community detection: community seed, liaison and direct recommender.

Definition 1: Community seed. seeds are the initial users in a community; it must be trust worthy and also have a large number of neighbors.

Definition 2: Liaison. They take charge of information dissemination among different communities. Liaisons belong to the overlapped area of different communities and they help people from different groups to communicate and share messages.

Thus the general form of our community structure discovery algorithm is as follows:

(1) Construct the adjacent map for network;

(2) Add intimacy value to each edge accords to the intimacy between vertices pairs and save this intimacy relation matrix;

(3) Calculate the fitness value for each vertex and build the fitness list;

(4) Find the community seeds, liaisons according to the chosen principle bellowed and divide the whole network.

In fact, it appears that the last two steps are the most important features of our algorithm as far as getting satisfactory results are concerned. As our studies mentioned above, once the community seed and the liaisons are found, the exact community structures will be very clear.

Seeds in a weighted network do not only need a large number of adjacent neighbors but also need to be trusted. We suppose that there is a network N, A_N and T_N are separately N's adjacent matrix and intimacy matrix, in that way, we use formula (2) to measure vertex i's average value of being trusted.

$$C_i = \frac{\sum_{j=1}^{n} T_{i,j}}{\sum_{j=1}^{n} A_{i,j}} \quad (2)$$

To better weigh the vertices' position in the network, in other words, to set a criterion to rank the vertices, we set a fitness function.

$$F_i = \alpha \times \sum_{j=1}^{n} A_{i,j} + \beta \times C_i \quad (3)$$

As we can see, fitness value is affected both by the number of adjacent neighbors and the average intimacy value. α and β decide the weight proportion of adjacent neighbors and the intimacy value. To distinguish the community seed and the liaison, we also set a minimum intimacy threshold ε which will be showed how it works in the following part.

Community seeds and liaisons are chosen as below shows.

All vertices are sorted in decreasing order of fitness which constitutes a list L_{fit}. Community seeds set *S* and liaison set L are both initialized to empty. Vertices in L_{fit} are checked in turn from the beginning to the end of the list. Calculate the similarity between free vertex *i* and the existing community *SN* , if similarity*(i, SN)*< δ and C_i > ε , it becomes a new seed and is added to *S*; else if C_i < ε , i becomes a liaison and is put into L; if there are many existing communities, and only similarity(SN,i)> δ, *i* becomes a member of *SN*, else if both similarity(SN1,i) > δ and similarity(SN2,i) > δ, we mark i as liaison and put i into the community with which it has larger similarity; Iteratively calculate the similarities until the end of the list. At the end, take out of the vertices in L and put them into suited communities.

The similarity between community *SN* and vertex *i* is calculated as formula (4) which is enlightened by CF.

$$Sim(SN,i)=cos(SN,i)=\frac{V_S \cdot V_i}{\sqrt{V_S^2}\sqrt{V_i^2}} \quad (4)$$

We can also use the Pearson coefficient to measure the relation similarity between free vertices and existing communities. For the simple networks we studied in this paper, we set the minimum similarity threshold as 1/n(n is the size of network). The flow chart of our algorithm is as figure 3 shows.

In the next part of our paper, we demonstrate the efficiency and expandability of our algorithm with a number of examples and show that our algorithm can be reliably and sensitively extract community structures from both artificially generated and real-world networks with known community structures. Besides, we also prove how our algorithm can be used to analyze networks whose structure is otherwise difficult to comprehend. The networks studied include a collaboration network of scientists, in which our method allows us to generate similarity comparison diagrams of the whole network.

III. IMPLEMENTATION

In theory, the description of the preceding section completely define the methods we consider in this paper, but in practice there are a number of subtleties to their

implementation that are important for turning the description into a workable computer algorithm.

```
Begin RACDA
    Initialize;
    Take out the first member in Lfit and put it into S;
    for i=1:n
        for j=1:k
            Calculate the Similarity value between vertex i and community SNj;
            If similarity(i,SNj) is higher than the minimum similarity threshold
                Calculate the trust value of vertex i;
                If Ci is higher than the minimum intimacy value
                    i becomes a seed;
                else i becomes a liaison
            else i becomes a member of community SNj
        end
    end
End RACDA
```

Figure 3. The detailed procedure of RACDA.

Essentially most of the work in this algorithm is in the calculation of the similarity between free vertices and the existing communities; the job of choosing appropriate α and β for fitness function becomes the most pressing matter of the moment for RACDA. Figure 3 uses pseudocode to show the general flow of our algorithm.

A. Undirected, weighted networks

This algorithm dynamically set the number of communities in a complex network accords to the community seeds. To illustrate the fundamental principles and running results of our algorithm, we constructed a small weighted network *N* like Figure 4 shows. The adjacent matrix and trust matrix of this network are showed below. From Figure 4 we can see that each edge has a weight to label the intimacy between vertices.

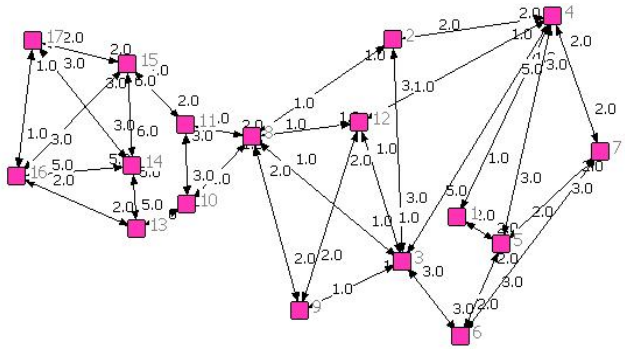


Figure 4. A weighted network

During the course of study we find that most scholars pay much attention to the degree of nodes which results in the misleading of communities. For example, vertex 8 will becomes a member of the left side community and plays a core role in it if we don't think about the weight on edges. However, we find that the values on edges that contact vertex 8 to others is very small which means vertex 8 is not very trust worthy. In other words, vertex 8 is a liaison instead of a seed. Here we show the adjacent matrix and the intimacy matrix of N, and present the results of RACDA. We use A_N to represent the adjacent matrix of Figure 4, T_N represent the trust matrix.

$A_N =$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	1	1	1	0	1	0	1	1	0	0	1	0	0	0	0	0
4	1	1	1	1	1	0	1	0	0	0	0	1	0	0	0	0	0
5	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
6	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0
7	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
8	0	1	1	0	0	0	0	1	1	1	1	1	0	0	0	0	0
9	0	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	0
10	0	0	0	0	0	0	0	1	0	1	1	0	1	0	0	0	0
11	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0
12	0	0	1	1	0	0	0	1	1	0	0	1	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	1	0
14	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
16	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
17	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

$T_N =$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	3	2	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	3	0	5	0	3	0	1	1	0	0	1	0	0	0	0	0
4	1	2	5	0	3	0	2	0	0	0	0	1	0	0	0	0	0
5	2	0	0	3	0	2	2	0	0	0	0	0	0	0	0	0	0
6	0	0	3	0	2	0	3	0	0	0	0	0	0	0	0	0	0
7	0	0	0	2	2	3	0	0	0	0	0	0	0	0	0	0	0
8	0	1	1	0	0	0	0	0	2	1	2	1	0	0	0	0	0
9	0	0	1	0	0	0	0	2	0	0	0	2	0	0	0	0	0
10	0	0	0	0	0	0	0	1	0	0	3	0	5	0	0	0	0
11	0	0	0	0	0	0	0	2	0	3	0	0	0	0	0	0	0
12	0	0	1	1	0	0	0	1	2	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	5	0	0	0	5	0	2	0
14	0	0	0	0	0	0	0	0	0	0	0	0	5	0	6	5	3
15	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	3	2
16	0	0	0	0	0	0	0	0	0	0	0	0	2	5	3	0	1
17	0	0	0	0	0	0	0	0	0	0	0	0	0	3	2	1	0

We set α =0.6 and β =0.4, the fitness table of Figure 4 is as Table I shows.

TABLE I.
THE FITNESS OF VERTICES IN NETWORK *N*

ID	F	ID	F	ID	F
1	4.4668	7	5.2	13	5.32
2	4.7	8	5.6284	14	5.88
3	5.9716	9	4.6	15	5.4
4	5.9716	10	5	16	5.24
5	5.08	11	4.8	17	4.7
6	4.9	12	4.76		

Formula (5) is used to set the minimum trust threshold.

$$\varepsilon = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} T_{i,j}}{2\times|E|} \qquad (5)$$

Formula (5) tells us that threshold ε is the average value of weight on each edge, for network *N*, ε = 2.45 and the minimum similarity threshold is 1/17. We use ○, △ and ◇ to separately represent the community seed, ordinary user and liaisons in a community. Figure 5

shows the dividing results of Figure 4 and Figure 6 is the output of RACDA.

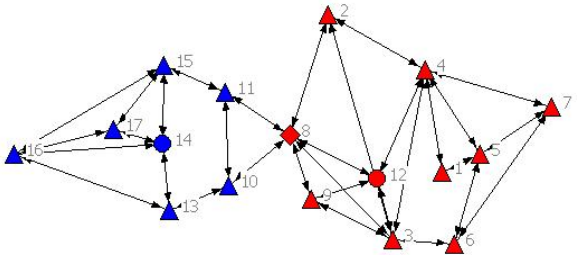


Figure 5. Dividing results of figure 4.

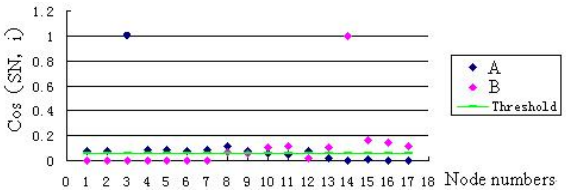


Figure 6 . The output of RACDA

The topology structure of Figure 5 is too simple to demonstrate the effectively of RACDA. It has no single vertices on the margin of different communities. Here we use another real-world network who has a little more complex relationship, the *Les Miserables* dataset. Using the list of character appearances by scene complied by Knuth[22], the network was constructed in the situation that the vertices represent characters and the edge between two vertices represent coappearance of the corresponding characters in the same one or more scenes, the value on edges represent the times of their coappearance in the same scene. Generally, there are six most important people on the network: Jean Valjean(vertex 11), detective Javert(vertex 27), father Bishop Myriel(vertex 0), grisette Fantine(vertex 23) and her daughter Cosette(vertex 26). 77 vertices and 508 edges are showed in the network altogether like Figure 7 shows.

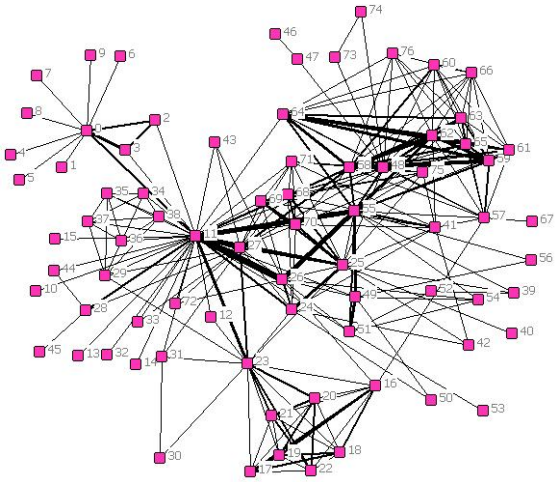


Figure 7 . The topology of Les Miserables.

From Figure 7 we can easily get the central vertices in *Les Miserables*. Compared vertex 43 and 72 we can see that their roles are different, and the only difference between 43 and 72 is that the intimacy between 43 and 11 are much stronger than the intimacy between 72 and 11 while the intimacy between 72 and 26 are stronger than 43 and 26. Figure 8 shows the dividing result of G-N to *Les Miserables*. Figure 9 illustrates the differences between community seeds and liaisons, although vertex 27 has a lot of neighbors, the intimacy between 27 and its neighbors is rather distant which limits 27 to a liaison rather a seed. Another difference between figure 9 and 10 is the situation of vertex 16 and 57, in figure 10 we can see that the relationship in communities is much closer than the relationship between different communities.

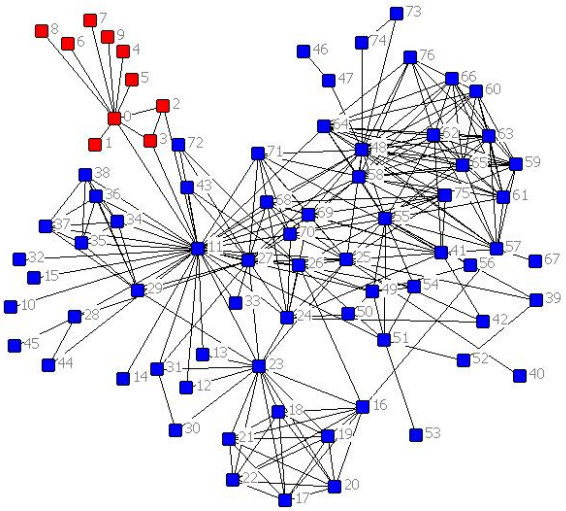


Figure 8 . The results of G-N to Les Miserables.

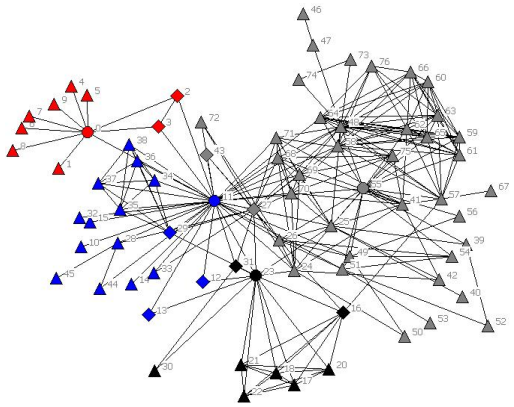


Figure 9 . Dividing results of RACDA to unweighted Les Miserables

Here we set α =0.6 and β =0.4 to calculate the fitness for vertices in *Les Miserables*. Figure 9 is the dividing result of unweighted *Les* while Figure 10 represents the dividing result to weighted *Les Miserables*.

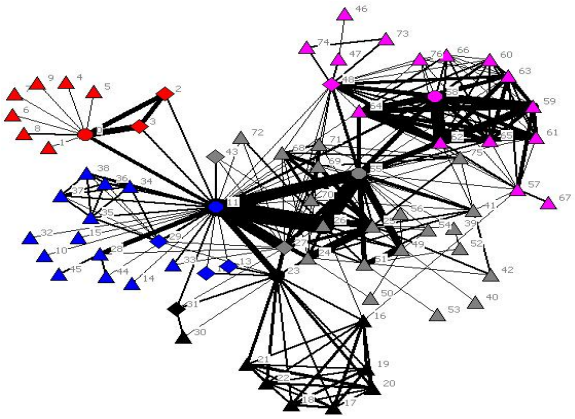


Figure 10 . The dividing result of RACDA to weighted Les Miserables.

On the procedure of clustering, we use intimacy relation-weighted community eigenvectors and get exciting results. If we do not add intimacy values to edge, some vertices who located at the margin area of different communities will be divided into wrong groups. Such as vertex 2 and 3, because vertex 11 is much more central than vertex 0 and it also has much more sophisticated relationships, which result in V_3 and V_2 fall into the community that vertex 11 in. In that way, vertices 2 and 3 were divided into the wrong community. However, if we use intimacy weighted eigenvectors, the denominator of similarity between vertex 3 and community A(vertex 11 lead) becomes very small while the denominator of similarity between 3 and community B(vertex 0 lead) stays almost the same. Taking both the adjacent relationship and the intimacy between pairs into consideration, we get exactly what we want.

B. Undirected,unweighted networks

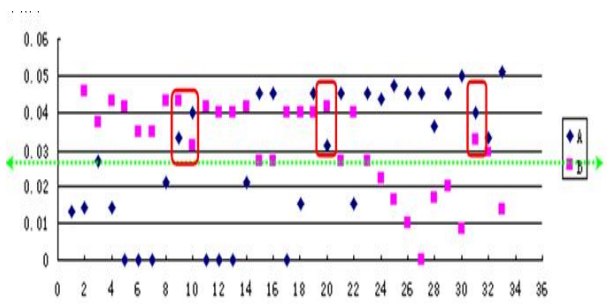


Figure 11 . Output of RACDA to Zachary club[23].

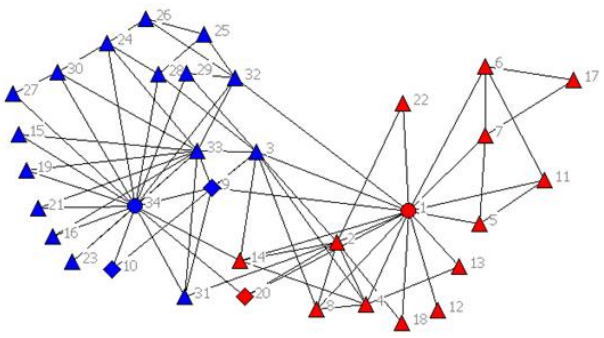


Figure 12 . Dividing result of RACDA to Zachary Club

Figure 11 is the output of RACDA to Zachary Club and Figure 12 is the dividing result. There is no weight of intimacy in unweighted networks, so we set $\alpha=1$ and $\beta=0$. However, we can see from Figure 11 that vertex 10 actually community A's inner member. The reason for this mistake is that we calculate the similarity only on the basis of whether the edge exist or not while lacking weight for edges to measure which edge plays more important roles on the network and which edge should be given a prior consideration. During some following experiments we prove that this mistake can be avoid if we adjust the community eigenvector.

IV. COMPARISON

Here we show the comparison results of modularity on different real-networks of various kinds of community discovery algorithms.

TABLE II.
THE COMPARISON OF DIVISION RESULTS OF DIFFERENT CDAS

Name	Football	Karate Club	Dolphins	Les Miserables
GN	0.377	0.395	0.381	0.082
Fast G-N	0.376	0.360	0.379	0.074
Polish	0.327	0.358	0.226	0.107
RACDA	0.484	0.401	0.379	0.486

From Table II we know that RACDA gets the highest modularity on all kinds of social networks, especially on complex networks. Fast G-N has smaller complexity while sacrifices the division accuracy.

The results of RACDA showed above use unweighted edges in the network which means the fitness function is as formula (6) shows.

$$F_i = \sum_{j=1}^{n} A_{i,j} \tag{6}$$

Under this situation, the degree of vertices becomes the only element to measure the centrality of vertices.

The primary remaining difficulty with our algorithm is how to use modularity to test the quality of our algorithm. For weighted networks, link-in and link-out numbers are not the only standard to judge the cohesion level of a community, but also the weight of edges are also needed to be taken into consideration, especially edges between different communities. Besides, the methods to define the minimum similarity threshold and the minimum trust threshold need to be optimized as well. Most networks with complicate topologies and indistinct levels have their own characteristics and we should set special values for them. However, a better approach would be to find some improvement in the algorithm itself to optimize the results of clustering.

V. CONCLUSION

In this paper, we describe a new algorithm for social community discovery, the task of extracting the natural

community structures from networks of vertices and edges. This is a problem long studied in computer science, applied mathematics, physics, and the social science. Especially with the development of SNS, community discovery is playing a more and more important role in E-commence and SNS websites. Whether from the viewpoint of social network service or network security, community discovery would become a hot-topic for the next development of complex adaptive system. We believe the methods described here give a good solution to community discovery. Our algorithm is defined by two crucial features. First, we use three labels to distinguish the users in the same community which provides a promise for personalized recommendation. Besides, the differentiation also makes it possible to give special protections for different users which enhance the robustness of the network. Second, our methods include a recalculation step until the vertices are all put into the right communities. This step, which determines the ascription of vertices, turns out to be of primary importance to the success of our algorithm. Without the similarity recalculation, this algorithm fails miserably at even the simplest clustering tasks.

REFERENCES

[1] M. E. J. Newman. "Finding Community Structure in Networks Using the Eigenvectors of Matrices." Physics. 2006, 1–22.

[2] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, Complex networks: "Structure and Dynamics." Physics Reports 424, 2006, 175–308.

[3] Leicht E A, Newman M E J. "Community Structure in Directed Networks." Physical Review Letters, 2008, 100:118703.

[4] S.N.Dorogovtsev and J.F.F.Mends, "Evolution of Networks: From biological Nets to the Internet and WWW". Oxford University Press, Oxford, 2003.

[5] G.W. Flake, S.R.Lawrence, C.L.Giles, and F.M.Coetzec, IEEE computer 35, 2002, 66.

[6] Rosvall M, Bergstrom C T. "An Information-theoretic Framework for Resolving Community Structure in Complex Networks." PNAS, 2007, 104(18): 7327-7331.

[7] A.Broder, R.Kumar, F.Maghoul, P.Raghavan, S.Rajagopalan, R.stata, A.Tomkins and J.Wiener, Computer. Netw. 2000, 33,309.

[8] B. Hu, X.Y. Jiang, J.F. Ding, Y.B. Xie, B.H. Wang. "A Weighted Network Model for Interpersonal Relationship Evolution." Physica A. 2005, 353:576-594.

[9] C.Moore and M.E.J.Newman, "Epidemics and Percolation in Small-world Networks." Phys. 2000, Rev. E61, 5678-5682

[10] LIU Ji, DENG Gui- shi. "A Collaborative Recommendation Method Based on User Network Community with Weighted Spectral Analysis". Journal of Dalian University of Technology. 2010, 50(3): 438-442.

[11] Cun-rui， Xiao-dong，LIU Xiang-dong，LI Zhi-jie. "A Physical Community Discovery Algorithm." MICROELECTRONICS&COMPUTER. 2010. 27(9): 33-36.

[12] M.E.J. Newman. "Scientific Collaboration Networks: II. Shortest paths, Weighted Networks, and Centrality." Phys. 2004, Rev. E64, 016132.

[13] M.E.J.Newman. "Mixing Patterns in Networks." Phys. 2003, Rev. E67, 026126.

[14] "Research of the Present Situation of Operation and the Future Tendency of SNS Website." http://media.people.com.cn/GB/22114/119489/140165/8454258.html（2009）

[15] LUO Ze-bi, XIE Qink-sheng. "Collaborative Filtering Recommendation Algorithm Based on Web Data Mining." Journal of Guizhou University(Natural Science). 2009, 26:40-43

[16] Guo Yan-hong, DENG Gui-shi, LUO Chun-yu. "Collaborative Filtering Recommendation Algorithm Based on Factor of Trust." Computer Engineering, 2008, 34(20): 1-3

[17] DAI Ya-e, GONG Song-jie. "Collaborative Filtering Recommendation Based on Fuzzy Clustering in Personalization Services." Computer Engineering & Science, 2009, 31(4): 110-116

[18] Ashish Sureka and Pranav Prabhakar Mirajkar. "An Empirical Study on the Effect of Different Similarity Measures on User-Based Collaborative Filtering Algorithms." Springer-Verlag Berlin. 2008, LNAI 5351: 1065–1070.

[19] Ahn, H.J. "A New Similarity Measure for Collaborative Filtering to Alleviate the New User Cold-starting Problem." Information Sciences: an International Journal. 2008, January.

[20] FAN Cong-xian XU Ting-rong FAN Qiang-xian. "Research and Improved Algorithm of HITS Based on Web Structure Mining." Computer Information. 2010, 26:160-162.

[21] M.E.J. Newman, M. Girvan. "Finding and Evaluating structure in networks." Physics Review[C]. 2004, E69, 026113

[22] D.E.Knuth, The Stanford Graphbase: "A Platform for Combinatorial Computing". Addison –Wesley, 1993, Reading, MA.

[23] Zachary W W. "An Information Flow Model for Conflict and Fission in Small Groups." Journal of Anthropological Research, 1970, 33:452–473.

Ruixin Ma, (1975--). Lecturer of Dalian University of Technology. Research area: E-commence, community discovery and swarm intelligence.

Guishi Deng, (1945--). Professor of Dalian University of Technology. Research area: E-commence, decision analysis and analysis of complex system.

Wang Xiao, (1988--). Ph.D. candidate at Institute of Automation, Chinese Academy of Science. Research area: E-commence, community discovery and swarm intelligence.

A Survey on Particle Swarm Optimization Algorithms for Multimodal Function Optimization

Yu Liu[1], Xiaoxi Ling[1,2], Zhewen Shi[1], Mingwei Lv[1], Jing Fang[1] and Liang Zhang[1]
[1]School of Software, Dalian University of Technology, Dalian, China
[2] Civil Aviation Flight University of China, Guanghan, P.R.China
Email: yuliu@dlut.edu.cn

Abstract— **Many scientific and engineering applications involve finding more than one optimum. A comprehensive review of the existing works done in the field of multimodal function optimization was given and a critical analysis of the existing methods was also provided. Several techniques in solving multimodal function optimization problems were introduced, such as clearing, deterministic crowding, sharing, species conserving and so on. And we summarized defects of existing algorithms: lacking of self-adaptive adjustment function, requiring setting some parameters according to different problems, lacking of unified theoretical and experimental system to guide algorithms design and not maintaining the diversity of swarm. Moreover, most of existing multimodal particle swarm optimization algorithms which include SPSO, MSPSO, ESPSO, ANPSO, kPSO, MGPSO, AT-MGPSO, rpso, and SDD-PSO were described and compared and advantages and disadvantages existing in these algorithms were pointed out. Therefore, some ideas to improve the performance of multimodal function optimization algorithms were proposed.**

Index Terms—**Multimodal Function Optimization, Evolutionary Algorithm, Particle Swarm Optimizer**

I. INTRODUCTION

Multimodal Function Optimization (MFO) is the problems which have more than one optimum, or exist only one global optimum and several useful local optima in feasible solution spaces. There are a large number of such problems existing in the real world, such as Rule Discovery in Data Mining, Neutral Network Integration, Fuzzy System, Optimization, Multi-solution Investigation and so on [32]. In order to offer a variety of options and more information to decision makers, multimodal function optimization algorithms try to find all these global optima and useful local optima. However, conventional optimization algorithms can only find one optimum randomly, and show ineffectiveness in solving such problems. Evolutionary Algorithm (EA) using population search instead of individual search in the conventional algorithms has obvious advantages and great probability to locate multiple optima simultaneously in feasible search spaces. Evolutionary Algorithm (EA) is the method that simulates natural biological activities and evolutionary process, establishes computation model and solves complex optimization problems. The search process only relies on fitness values, which is independent of gradient information. And that is widely used and not constrained by the differentiable and continuous limitations of problems, and has strong universality. Therefore, the application of Evolutionary Algorithm (EA) for solving multimodal function optimization problems, which has significantly theoretical value and practical significance, has aroused a widespread concern in the field of evolutionary computation and has become an important research field.

We can see the definition of Multimodal Function Problems from equation (1).

$$\min z = f(X) \text{ or } \max z = f(X),\ X = (x_1, \cdots, x_n) \in R^n \quad (1)$$

$\exists (X_1, X_2, \cdots, X_k)$ When $X^* = X_1, X_2, \cdots, X_k$, $\forall (X) f(X) \le f(X^*)$ or $\forall (X) f(X) \ge f(X^*)$, $s.t. \|X^*, X\| \le r_k$.

Where r_k is a parameter that represents the radius of the kth optimum solution's territory. $\|X^*, X\|$ represents the Euclidean Distance between X^* and X.

In this paper, the formulation of multimodal function optimization was given. And we gave a comprehensive review of the existing works done in multimodal function optimization and analyzed several popular techniques, such as clearing, deterministic crowding, sharing, species conserving and so on. We also summarized major defects of most existing multimodal function optimization algorithms suffered. Moreover, we compared most existing multimodal particle swarm optimization algorithms which include SPSO [24], MSPSO [25], ESPSO [26], ANPSO [30], kPSO [27], MGPSO [28], AT-MGPSO [29], rpso [32], and SDD-PSO [31].

II. NICHING TECHNIQUES

Genetic Algorithm (GA), which has developed completely during a long time, is the most representative algorithm in the evolutionary computing. New techniques and theories are all based on Genetic Algorithm (GA). In the standard Genetic Algorithm, due to the random of selection and reorganization, genetic drift exists in the population [1], which makes the population converge to an absorbing state and finds only one optimum. In order to solve multimodal function optimization problems,

doi:10.4304/jsw.6.12.2449-2455

researchers restrict the search behaviors of the standard Evolutionary Algorithm (EA), and introduce niching techniques, which divides the swarm into several small swarms by some means. Algorithms select and evolve in the entire population, and at the same time save the best individual in each niche. The algorithms not only maintain the diversity of individuals, but also retain a variety of high-fitness individuals, which prevents a high-fitness individual from filling with the entire population quickly and can find more than one global optimum or local optimum. The representative niching techniques are crowding methods, fitness sharing, and sequential niche technique and species conservation.

Crowding technique maintains the diversity of population by appropriate replacement policy. It is based on the theory that a variety of creatures have to compete for a variety of limited resources to survive in a limited living space. Cavichio [2] first proposed niche implement based on pre-selection in 1970: if offspring's fitness value is higher than the poor individuals of parents, it will replace the parent. In 1975, Jong [3] extended Cavichio's pre-selection mechanism and proposed standard crowding mechanism: select $1/CF$ (Crowding Factor) individuals from the population to form a temporary sub-population, and then compare the similarity which is determined by the number of matched alleles between new individuals and the individuals in the temporary sub-population. Then a new individual replaces the individual that has the greatest similarity with it. Although standard crowding technique improves the diversity of population, it cannot solve the situation of more than two peaks due to replacement deviation and shows a limited ability to maintain niche in multi-peak problems [4]. Malifoud [5] extended standard crowding technique by including competition between parents and offspring in the same niche and proposed Deterministic Crowding (DC): divide population whose size is N into $N/2$ pairs and each pair of parent individuals generates two offspring individuals through crossover and mutation, and offspring and parents compete to become members of next population by a competition league. Deterministic Crowding technique has solved some multi-peak optimization problems and shows strong ability to maintain niches. However, replacement policy of Deterministic Crowding (DC) can lead to loss of low-fitness niches. In order to overcome the defect, Mengshoel [6] proposed Probability Crowding (PC): offspring compete with similar parents and replace parents according to fitness values.

Fitness sharing is based on the theory that individuals are punished as the presence of other individuals in the survival space. That means individuals compete for limited resources to survive in the same niche. An individual's allocated resources decreases as the presence of other individuals and the individual selects the niche which has the maximum average of resources rather than which has the maximum of resources. Goldberg and Reihardson [7] firstly proposed a niching technique based on fitness sharing in 1987. The mechanism defined a sharing function and the niche radius controls the shape of sharing function as a constant. Sharing Function is the function on the similarity between two individuals (similarity of genotype or similarity of phenotype). When the similarity between two individuals is larger, the value of sharing function is larger; on the contrary, the value is smaller. This function determines each individual's sharing value. An individual's sharing value is equal to the sum of sharing function values between the individual and each of other individuals in the same population. Then each individual's fitness is adjusted according to its sharing value. After that, the algorithm can carry out selection operation based on the new sharing value to maintain the diversity of the population and create the niche's evolutionary environment in the process of population evolution. This model solved multimodal function optimization problems successfully and became a popular niching technique. Researchers proposed a number of improved algorithms in order to improve the effectiveness of fitness sharing genetic algorithm. Petrowski [8] proposed clearing improved fitness sharing technique. Clearing divides the population into q niches according to clustering analysis. However, due to the concept of limited resources, each niche only retains k better individual's fitness value and other individuals' fitness value is set to 0. Each niche only needs to compute sharing value for k better individuals, which reduces the cost of sharing technique's implementation. Dynamic Niche Clustering [9] (DNC) regards each individual in the initial population as a niche, and merges, decomposes and creates niches dynamically by fuzzy clustering analysis, and calculates sharing value in each niche.

Sequential Niche Technique [10] runs Genetic Algorithm (GA) repeatedly for the same problem and maintains the optima which have been found at the end of each run. In order to reduce the probability of repeatedly visiting the same optimum which has been found, this algorithm uses a similar method of calculating fitness sharing to reduce all the solutions' fitness values within the radius of optima that have been found during the next run.

Species Conserving [11] sorts the individuals according to fitness values at each generation. First, the algorithm selects the best individual as the first species seed of seed collection, and then each individual is compared with individuals in the seed collection sequentially. If the distance between an individual and all the species seeds is longer than specified radius, this individual will be added into seed collection as a new seed. The algorithm will not end until the worst individual has compared with the individuals in seed collection and formed final seed collection, which conserved to the next generation.

In order to solve multimodal function optimization problems, on the one hand, algorithms have to maintain the diversity of the population in order to converge on multiple solutions. On the other hand, algorithms must have better global search capability and can get rid of insignificant local optima and find better optima rapidly. They are not only essential, but also restricted to each other. Genetic Algorithm's defect of weak local search

capability makes it have better global search capability and can find vicinity of optima rapidly. However, if Genetic Algorithm (GA) wants to improve the solutions' qualities, it needs expensive computation cost. There are two kinds of thoughts to improve the search capability of Genetic Algorithm (GA). One is combining local search methods with genetic algorithm (GA) to overcome genetic algorithm's defects. Another one is using new principles of evolutionary search. Researchers try to use Differential Evolution [12], Artificial Immunity [13-14], Evolution Strategy [15] and Particle Swarm Optimizer (PSO) [17-25] to solve multimodal function optimization problems.

III. Multimodal Particle Swarm Optimization

Particle Swarm Optimizer (PSO) [16] is an evolutionary algorithm proposed by Kennedy and Eberhart in 1995. Particle Swarm Optimizer (PSO) imitates population's social behavior instead of relying on individual law of natural evolution. The existence of individual and individual, individual and population interaction and mutual influence behavior shows that sharing information exists in the population. Individuals rely on simple rules of particles to produce complex social behaviors and get better performance through sharing information and constant interaction. Particle Swarm Optimizer (PSO) develops very rapidly and becomes one of hot research field in evolutionary computation. Especially in the continuous variable function problems, Particle Swarm Optimizer (PSO) shows great potential and outstanding performance and stands out from a number of evolutionary algorithms.

Sharing information based on swarm intelligence provides a new thought for algorithm design. Particle Swarm Optimizer (PSO) is still in its infancy and most studies focus on the improvement of algorithm's performance. The use of Particle Swarm Optimizer (PSO) to solve multimodal function optimization problems has just started. Kennedy [17] introduced cluster technique to solve multimodal function optimization problems. Li et al [18] introduced sharing function used in Genetic Algorithm (GA) into Particle Swarm Optimizer (PSO). Parsopoulos et al [19] made use of compression and stretching techniques to solve multimodal function optimization problems. Brits [20] proposed NichePSO by using sub-swarm. Ozcan [21] improved the performance of NichePSO by using fanaticism mechanism and climbing mechanism, however, this algorithm need mutually set parameters associated with niche. Seo [22] developed a new strategy of selecting attractor *gbest* to solve multimodal function optimization problems; however, the algorithm needs to set the number of peaks artificially. Li [23] proposed multimodal function optimization based on Fitness Euclidean-Distance Ratio (FED). Particles select the individual's *gbest* whose fitness value has changed the most greatly in unit distance according to Fitness Euclidean-Distance Ratio (FED) as the attractor when particles update velocities and the algorithm has the similar problems with sharing function technique, which is that the algorithm needs to compute N^2 (N is the population size) Fitness Euclidean-Distance Ratio. When optimizing complex problems, the algorithm needs a larger population size and the computation cost will increase sharply.

Most of existing multimodal particle swarm optimization algorithms are based on species conservation and fitness sharing. SPSO [24], MSPSO [25], ESPSO [26], ANPSO [30], kPSO [27], MGPSO [28], AT-MGPSO [29], and rpso [32] are based on species conservation. SDD-PSO [31] is based on fitness sharing.

Algorithms based on species conservation can be divided into two classes, depending on niching parameters and parameters independent. SPSO and MSPSO depend on niching parameters. Users need to prespecify the niching parameter which determines how large a niche is. Depending on niching parameters is the greatest disadvantage of these algorithms. Users have to set different radius according to different problems. And this restricts these algorithms widely used in the real world. Although ANPSO, MGPSO and AT-MGPSO also need niching parameters, these algorithms adaptively determine niching parameters at each generation. ANPSO uses the intrinsic nature of the particles to converge on optima and create niches when they do so. And MGPSO and AT-MGPSO determine the size of niches according to the feasible solution space. ESPSO doesn't depend on niching parameters, but it introduces three new parameters s, m and δ. The first two have been shown to be robust across the test functions. The last is problem dependent; however, it is intuitive and easy to set. kPSO is also independent of niching parameters. However, depending on parameter c and the number of steps between clustering limits the algorithm widely used in the real world. rpso doesn't depend on any niching parameter. And it shows that PSO with ring topology is able to induce stable niching behaviors.

Li proposed a Species-based Particle Swarm Optimization (SPSO) [24] in 2004. The algorithm needs to set a niching parameter r. At each generation, SPSO sorts all the particles descendingly by the current fitness values. Then the algorithm determines whether two particles are in the same niche by calculating the Euclidean Distance between them. If the distance is smaller than r, the two particles are in the same niche. If the distance is larger than r, they are in different niches. After the niches divided, the algorithm updates each particle's *gbest* in each niche by the best particle's *pbest* in this niche. Experiments show that SPSO has good performance in solving several standard multimodal test functions. However, depending on the parameter r which determines the size of a niche is the greatest weakness of SPSO. If r is too small, particles probably trap into local optima, or the algorithm may miss some global optima.

IWAMATSU and Masao [25] proposed a modified Particle Swarm Optimizer (PSO) called Multi-species Particle Swarm Optimization (MSPSO) to locate all the global optima of multimodal function optimization problems. MSPSO has the same idea as SPSO; however,

MSPSO sorts all the particles descendingly by their personal best fitness values at each generation.

Bird and Li proposed an adaptively niching parameters Particle Swarm Optimization algorithm (ANPSO) [30] in 2006. Due to most of existing algorithms need to set niching parameters to determine a niche's size. This greatly reduces the algorithms' usefulness and effectiveness. ANPSO adaptively determines niching parameters at each step. The method uses the intrinsic nature of the particles to converge on optima and create niches when they do so. When a particle discovers a local peak, its velocity will reduce and it will explore the area closely. If multiple particles explore the same area, it is likely that it is a peak of interest, so a niche is created with those particles as its initial members. Any other particles that later converge on the peak will also join the niche. In step 2, an undirected graph *g* is created containing a node for each particle. Initially there are no edges between any of the nodes. Step 3 finds every set of particles that have been close to each other for at least 2 steps. The algorithm finds the pairs of particles that are within *r* of each other. A counter is maintained for each pair – if the pair has been close for two or more steps, a niche is formed. After above three steps, all the particles are either in a particular niche, or are not in any niche. For each niche, using the best particle's *pbest* updates other particles' *gbest*. The unniched particles are placed in a von Neumann Neighborhood which is updated at each step. Although ANPSO has good performance in handling some standard multimodal test functions, it has poor performance in solving high dimensional multimodal problems.

Bird and Li proposed an enhanced version of SPSO, ESPSO [26] in 2006. ESPSO enhances SPSO by greatly increasing the robustness of the niching parameter – to point that the algorithm is still effective even if it isn't used at all. ESPSO includes Detecting Convergence, Preventing Future Interactions and Removing Duplicate Species into PSO. It introduces three new parameters *s*, *m* and δ . The first two have been shown to be robust across the test functions. The last is problem dependent; however it is intuitive and easy to set. However, depending on the three parameters affects the algorithm's effectiveness and efficiency.

Passaro and Starita [27] proposed a new algorithm called k-means-based PSO (kPSO) to niching in PSO based on clustering particles to identify niches in 2008. kPSO employs the standard k-means clustering algorithm and Bayesians Information Criterion to adaptively identify the number of clusters. Through solving a set of multimodal test functions, kPSO shows better performance than some other existing algorithms. Although kPSO is a competitive solution in solving multimodal function optimization problems, the algorithm needs to set parameter *c* and the number of steps between clustering. Depending on parameter *c* and the number of steps between clustering limits kPSO widely used in the real world.

Seo et al [28] proposed a new algorithm solving multimodal function optimization problems called Multi-grouped Particle Swarm Optimization (MGPSO). The algorithm gives every group a territory to avoid overlapping of discovered solutions. To encourage the individual particle they proposed the concept of repulsive velocity, located in territory of other group, to escape from the other group's territory in more efficient manner. In this algorithm, if a particle intrudes other group's territory, the group will push the particle out from its own territory by updating the particle's velocity according to equation (2).

$$V_{ij}^{k+1} = wV_{ij}^{k} + C_1R_1\left(pbest_{ij}^{k} - X_{ij}^{k}\right) + C_2R_2\left(gbest_{i}^{k} - X_{ij}^{k}\right) + C_3R_3\left(X_{ij}^{k} - gbest_{m}^{k}\right) \quad (2)$$

Where X_{ij}^{k} and V_{ij}^{k} are the position and velocity of the *j*th particle in the *i*th group at *k*th iteration. w is inertia weight and is usually decreasing linearly from 0.9 to 0.4 throughout the simulation. $pbest_{ij}^{k}$ is the *i*th group's *j*th particle's personal best (*pbest*) at the *k*th iteration. $gbest_{i}^{k}$ is the *i*th group's global best (*gbest*) at the *k*th iteration. $gbest_{m}^{k}$ is the global best (*gbest*) of intruded group. C_1 and C_2 are acceleration factors which determine the relative pull for every particle toward personal best (*pbest*) and global best (*gbest*). C_3 is repulsively coefficient. However, it has a zero value if the *j*th particle in the *i*th group doesn't intrude other group's territory. R_1, R_2 and R_3 are random numbers between 0 and 1. The forth term of the equation is a repulsive velocity component. And it is used to push the particle out from the territory of other group's intruded by the particle. The algorithm protects all the groups' territories from intruding by other group's particles. In MGPSO, all the groups have the same size of territories that means they have the same radius, and the radius decreases linearly in a decreasing order throughout the iteration. However, if the radius became too small before sufficient convergence level, some groups cannot find their own solutions and wandered around other groups' solutions. To overcome this defect, they proposed another algorithm based on MGPSO called Auto-tuning Multi-grouped Particle Swarm Optimization (AT-MGPSO) [29] in 2008. In AT-MGPSO, competition mechanism is invited and all the groups have different radius. When two groups' territories overlapped, the winner group which has the higher global fitness remains and the radius is increasing by dividing it by 0.95. And the loser group is expelled and reinitialized out of the existing groups' territories and searches other solutions. The algorithm is based on the idea that is such that a group with a broad scope of influence and high fitness has higher probability to be invaded by other groups. Although AT-MGPSO has good performance in handling simple multimodal test functions, it needs to set the niching parameter which determines how large a niche is. In this algorithm, *r* is set to 5% of the whole solution space. However, for simple problems, AT-MGPSO can get good performance, for complex problems, it cannot get so good performance. For example, for two-dimensional Shubert function, it has 18

global optima divided into 9 clusters; the distance between two optima in the same cluster is very small. And the niching parameter in AT-MGPSO is much larger than that distance, so we may miss some global optima.

Li proposed a new niching Particle Swarm Optimization based on ring topology in 2010 [32]. The algorithm doesn't depend on any niching parameter. He has demonstrated that the PSO algorithms with ring topology are able to induce more stable niching behavior. The PSO algorithms with overlapping are able to locate multiple global optima, given a reasonably large population size, whereas the PSO algorithms with a non-overlapping ring topology can be used to locate global as well as local optima, especially for low dimensional problems. Experiments show that the PSO algorithms with a ring topology can provide comparable or better, and more consistent performance, than some existing niching PSO algorithms. Even with a comparable or smaller population size, the proposed algorithms can outperform a niching algorithm using a fixed niche radius, in terms of success rate and the actual number of global optima found. Most importantly, one major advantage over existing niching algorithms is that no niching parameters are required. This should pave the way for more widespread use of this kind of niching algorithms in real-world applications. The algorithm is the first attempt showing that PSO algorithms with ring topology are able to induce stable niching behavior. The algorithm suggests that local memory and slow communication topology are two key elements for the success.

Most of algorithms based on fitness sharing introduce new mechanism to maintain diversity of swarm in order to locate all the global optima of multimodal functions. In SDD-PSO [31], a mutation operator is introduced to prevent premature convergence in high dimensional functions. This algorithm results in a superior performance and robustness to all other parameter configurations tested for some standard test functions.

Worasucheep [31] proposed a modified PSO with two techniques: a mutation operator to increase swarm diversity for high-dimensionality and an improved mechanism to detect and resolve the stagnation once it is found. In that paper, stagnation detection and dispersion (SDD) mechanism was modified and enhanced. A mutation operator is introduced to prevent premature convergence in high dimensional functions. Before the velocity and position update in each generation, the mutation method works as follow. For each particle, if a randomized value is below a mutation probability (*mp*), the particle's position is re-randomized within the range of each dimension. This applies to every dimension of the mutated particle. A few internal parameters for swarm dispersion are adjusted in order to increase the strengths of dispersion effects. This enhancement is more essential for asymmetric initialization (than in symmetric initialization). It should be beneficial for high-dimensionally as well. This configuration is based on a sensitivity analysis of the results from the preliminary experimentation using various parameter configurations on a set of test functions. This algorithm results in a superior performance and robustness to all other parameter configurations tested for some standard test functions. However, this algorithm doesn't have good local search capability.

Due to evolutionary algorithms' bionic objects and mechanisms are different, algorithms' search mechanism, optimization efficiency and scope are different. In genetic algorithm (GA), the generation of new individuals is mainly driven by crossover; however, mutation is only as a supplement and cannot afford a decisive role. Therefore, genetic algorithm (GA) has better global search capacity and can locate the vicinity of optimal solution quickly; however, crossover hardly makes individuals have better local search capacity, and only rely on reducing the population converge to refine the solution. So genetic algorithm (GA) has weak capacity of local search and has low search efficiency. In evolution strategy and artificial immunity algorithms, the generation of new individuals is mainly driven by mutation. Individuals have strong local search capacity; however, due to lacking of interactions between individuals and worse global search capacity, individuals easily fall into valueless local optima. Particle Swarm Optimization's (PSO) powerful search capacities are from the advanced search driving principle (pattern of generating new individuals). Particles complete self-learning function and social-learning function by use of sharing information and have a good balance between global search capacity and local search capacity and have high search efficiency. Particle Swarm Optimizer (PSO) has better search capacity than other algorithms and has absolute advantages in the continuous variable function optimization problems. Furthermore, Particle Swarm Optimizer (PSO) has a distinct biological background of swarm intelligence. Sharing information provides a new thought for algorithms design. And algorithm's performance still has a large space to further improve. Species conservation applied on Particle Swarm Optimizer (PSO) has better performance than that applied on Genetic Algorithm (GA) based on previous research experiments, which shows Particle Swarm Optimizer's (PSO) advantages and great probability. However, Particle Swarm Optimizer (PSO) doesn't have good performance and high success rate in solving high dimensional and complex multimodal function optimization problems and most optimization algorithms based on Particle Swarm Optimizer (PSO) depends on prior knowledge of specified issues. And the algorithms can fail because of inappropriate parameters.

IV. Applications To Multimodal Function Optimization

In the real world, there are many multimodal function optimization problems about mathematical and project problems. Many advances in science, economics and engineering rely on numerical techniques for computing globally optimal solutions to corresponding optimization problems. These problems are extremely diverse and have many optimal solutions. For example, optimization problem about neural network's structure and weight, complex system's parameter, structure recognition,

economic modeling, neural networks training, image processing and engineering design and control and so on.

Scientific and engineering applications have many optimal solutions and regularly require algorithms to locate all the optimal solutions. This problem is NP-hard in the sense of its computational complexity even in simple cases. For example, nonlinear least squares problems, as well as feed forward neural network training and solving systems of equations [32], a comparatively simple task when only a few equations and unknowns are involved, grow notably more complex as the problem's dimensionality increases. The stochastic search algorithms are widely used in evolving artificial neural network (ANN) architecture and weights. As a rule, the best weights or architecture of an ANN are not exclusive. In fact, the different architecture of an ANN is very useful in different situations.

Multimodal function optimization is also widely used on magnetic. Interior Permanent-Magnet Synchronous Motor (IPMSM) [28] is a practical multimodal function optimization example which has four poles and 18 slots was selected. The objective of the optimal design was the maximum motor efficiency. However, if the value of the motor efficiency is similar, the solution with a lower winding temperature is preferred.

Multimodal function optimization is also widely used in the field of detecting intrusion [33]. The detecting intrusion technique based-rules are difficult to manage rule base and to establish statistical model. This technique also has higher false alarm rate and omission rate. However, multimodal function optimization algorithms can solve these problems.

In the field of cryptography [34], a number of hard and complex computational problems have been motivated. These problems have more than one optimum solution and the computational cost in solving them is very expensive. Such problems are the integer factorization problem related to the RSA cryptosystem; the index computation or the discrete algorithm problem related to the EI Gamal cryptosystem and so on. We can use multimodal function optimization algorithms to solve these problems in an efficient way.

V. Conclusions

Evolutionary Algorithm has unique advantage in solving multimodal function optimization problems. Though multimodal function optimization algorithms based on Evolutionary Algorithm have been developed for a long time and varieties of techniques appeared and were used successfully in application, the design and theoretical research of algorithms are far from perfect and there are many issues worthy for further study. For instance:

(1) Due to the algorithms are lack of self-adaptive adjustment function and some parameters need be set mutually according to problems, which limit the algorithms' flexibility and scope and make the algorithm difficult to apply to practical applications widely. For example: a niche radius relies on prior knowledge of specified issues and inappropriate value will make the algorithm fail; Fixed-size population can only maintain a certain number of equilibrium sub-population. For the specific applications, the algorithm is lack of estimation of population size parameters or the adaptively dynamic adjustment.

(2) Lack of unified theoretical and experimental system to guide algorithms design. It is difficult to analyze from theory and compare different algorithms' capacity of formation and maintenance niche.

(3) Niching techniques achieve the purpose searching multiple areas by increasing the diversity of population which is at the expense of decreasing local searching capability and convergence speed. If not improve the algorithm's search capability, though the algorithm can find multiple solutions, it need run a long time and even cannot find desirable solutions. And the algorithm will lose practical significance.

In the future, in order to solve the above problems, multimodal function optimization algorithms can introduce reinforcement learning mechanism to enhance the algorithms' search capacity. And we can use sharing information technique to make algorithms have a good balance between global search capacity and local search capacity. Other mechanisms can be introduced into multimodal function optimization algorithms to make the algorithms not rely on the prior knowledge of specified issues and self-learn the niching parameters. That can make the optimization algorithms have better practicality. We can study how to increase the search capability of small niches so that the performance of these niches will locate well with increasing dimensions. We can also develop techniques to adapt or self-adapt the population size, as the population size is a parameter that also needs to be supplied by users and focus on applying multimodal particle swarm optimization algorithms to tracking multiple peaks in a dynamic environment.

Acknowledgment

This work was funded by the Natural Science Foundation of China under grant No. 60803074 and the Fundamental Research Funds for the Central Universities (No. DUT10JR06).

References

[1] Mahfoud SW, "Genetic drift in sharing methods," IEEE World Congress on Computational Intelligence. 1994, pp. 67-72.

[2] Cavicchio D.J, "Reproductive adaptive plans," Proceedings of the ACM annual conference, 1972, pp. 60-70.

[3] De Jong and K.A, "Analysis of the behavior of a class of genetic adaptive systems," [Ph. D Thesis], Michigan: University of Michigan, 1975.

[4] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," *IEEE Transl. J. Magn. Japan*, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[5] SW Mahfoud, "Crowding and preselection revisited," Amsterdam Elsevier in Parallel Problem Solving from Nature, 1992, pp.27-36.

[6] Mengshoel OJ and Goldberg D E, "Probabilistic crowding: Deterministic crowding with probabilistic replacement," Proceedings of the Genetic and Evolutionary Computation Conference , 1999, pp. 409-416.
[7] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application, 1987, pp. 41-49.
[8] Petrowski A, "A clearing procedure as a niching method for genetic algorithms," Proceedings of IEEE International Conference on Evolutionary Computation, 1996, pp.798-803.
[9] Gan J and Warwick K, "Dynamic Niche Clustering: a fuzzy variable radius niching techniquefor multimodal optimisation in Gas," Proceedings of the 2001 Congress on Evolutionary Computation, 2001, pp. 215-222.
[10] Beasley D, Bull DR and Martin RR, "A Sequential Niche Technique for Multimodal Function Optimization," Evolutionary Computation, 1993, pp. 101-125.
[11] Li J.P., Balazs M.E., Parks G.T., et al, "A Species Conserving Genetic Algorithm for Multimodal Function Optimization," Evolutionary Computation, 2002, pp. 207-234.
[12] Thomsen R, "Multimodal optimization using crowding-based differential evolution," Proceedings of the 2004 Congress on Evolutionary Computation, 2004, pp. 1382-1389.
[13] LN De Castro and Timmis J, "An artificial immune network for multimodal function optimization," Proceedings of the 2002 Congress on Evolutionary Computation, 2002, pp. 699-704.
[14] LN De Castro and Von Zuben FJ, "Learning and optimization using the clonal selection principle," IEEE Transactions on Evolutionary Computation, 2002, pp. 239-251.
[15] Im C.H., Kim H.K., Jung H.K., et al, "A novel algorithm for multimodal function optimization based on evolution strategy," IEEE Transactions on Magnetics, 2004, pp. 1224-1227.
[16] Kennedy J and Eberhart R, "Particle swarm optimization," IEEE International Conference on Neural Networks, 1995, pp. 1942-1948.
[17] Kennedy J, "Stereotyping: improving particle swarm performance with clusteranalysis," Proceedings of the 2000 Congress on Evolutionary Computation, 2000, pp. 1507-1512.
[18] Li T, Wei C and Pei W, "PSO with sharing for multimodal function optimization," Proceedings of the 2003 International Conference on Neural Networks and Signal Processing, 2003, pp. 450-453.
[19] Parsopoulos KE and Vrahatis MN, "On the computation of all global minimizers through particle swarm optimization," IEEE Transactions on Evolutionary Computation, 2004, pp. 211-224.
[20] Brits R, Engelbrecht AP and van den Bergh F, "A Niching Particle Swarm Optimizer," Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning, 2002, pp. 692-696.
[21] Ozca and E.,Yilmaz M, "Particle Swarms for Multimodal Optimization," ICANNGA 2007, Part I, LNCS 4431, 2007, pp. 366–375.
[22] Seo, J.H., et al, "Multimodal Function Optimization Based on Particle Swarm Optimization," IEEE Transactions on Magnetics, 2006, pp. 1095-1098.
[23] Li X, "A multimodal particle swarm optimizer based on fitness Euclidean-distance ratio," Proceedings of Genetic and Evolutionary Computation Conference, 2007, pp. 78-85.
[24] Li X, "Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization," Proceedings of Genetic and Evolutionary Computation Conference, 2004, pp. 105-116.
[25] IWAMATSU and Masao, "Multi-Species Particle Swarm Optimizer for Multimodal Function Optimization," I IEICE transactions on information and systems, 2006, pp. 1181-1187.
[26] Bird S and Xiaodong Li, "Enhancing the robustness of a speciation-based PSO," IEEE Congress on Evolutionary Computation, 2006, pp. 843-850.
[27] Alessandro Passaro and Antonina Starita, **"**Particle Swarm Optimization for Multimodal Functions: A Clustering Approach," Journal of Artificial Evolution and Applications, 2008, pp. 1-15.
[28] Jang-Ho Seo, Chang-Hwan Im, Sang-Yeop Kwak, et al, "Multimodal Function Optimization based on Particle Swarm Optimization," IEEE TRANSACTIONS ON MAGNETICS, 2006, pp. 1095-1098.
[29] Jang-Ho Seo, Chang-Hwan Im, Sang-Yeop Kwak, et al, "An Improved Particle Swarm Optimization Algorithm Mimicking Territorial Dispute Between Groups for Multimodal Function Optimization Problems," IEEE TRANSACTIONS ON MAGNETICS, 2008, pp. 1046-1049.
[30] S. Bird and X. Li, "Adaptively choosing niching parameters in a PSO," Proceedings of the 8th annual conference on Genetic and evolutionary computation, Seattle, Washington, USA, 2006, pp. 3-10.
[31] C. Worasucheep, "A particle swarm optimization for high-dimensional function optimization," Electronics Computer Telecommunications and Information Technology (ECTI-CON), 2010 International Conference on Electrical Engineering, 2010, pp. 1045-1049.
[32] Xiaodong Li, "Niching Without Niching Parameters: Particle Swarm Optimization using a Ring Topology," IEEE Transactions on Evolutionary Computation, 2007, pp. 150-169.
[33] Lizhong Xiao, Zhiqing Shao and Gang Liu, "K-means Algorithm Based on Particle Swarm Optimization Algorithm for Anomaly Intrusion Detection," The Sixth World Congress on Control and Automation in Intelligent, 2006, pp. 5854-5858.
[34] E. C. Laskari, G.C. Meletiou, Y.C. Stamatiou, et al, "Evolutionary computation based cryptanalysis: A first study," Nonlinear Analysis, 2005, pp. 823-830.

A Policy-based Adaptive Web Services Security Framework

Bin Li, Lingjun Zhao, Junwu Zhu, Jun Wu
School of Information Engineering
Yangzhou University
Yangzhou, China
Email: lb@yzu.edu.cn, zljbox@gmail.com, jdkr@163.com, j_wu@vip.sohu.net

***Abstract*—Web services security has become a hot topic in the research of service oriented computing. This paper aims to study many pivotal technologies in the web services security. Firstly, a policy-based framework for adaptive web services security is proposed, with the policy concept, management mechanism and execution mechanism can be separated effectively, moreover, by management of user context and web services context, web services access control can adapt to the changed environment. Secondly, a policy description language called ReiT is given, ReiT is a declarative language based on the rules and ontology and can express the structural and non-structural knowledge. A mixed reasoning mechanism is proposed, the web service access control policy including the user context and web services context can be evaluated by the reasoner. Finally, a policy aware BDI agent to authorize the access control of the web services is presented, and a prototype system based on Java EE and Jade Agent platform is implemented, Simulation experimental results and an example demonstrate the security framework is feasible and effective.**

***Keywords*—Web Service Security, Policy, Context-awareness, Ontology, Agent**

I. INTRODUCTION

Web services, which are issued in recent years, are a newly web-oriented development and integration framework for distributed applications. Web services represent more loosely-coupled distributed application architecture. And they provide an effective way for application to cooperate in an open environment. To guarantee the web services normal operation, the access control of the web services is employed. Therefore, research on the security framework of the web services is imperative[1][2].

Up to now, there are a number of access control models that have been developed, such as Role-Based Access Control (RBAC)[3], Task-Based Access Control (TBAC)[4], Provision-Based Access Control (PBAC)[5], etc.

Unfortunately, the current security technologies of web services have three limits: one is the lack of flexible, extensible, and loosely-coupled. The other is the limit of the network adaptability and could not adjust according to the dynamic environment. As in open systems, web services environment and their own states are ever changing that need to deal with the dynamic characters such as states and behaviors. Three is the lack of the autonomous. Web services themselves have no ability of proactive interaction. And they can not adjust their own states and behaviors.

To address the above mentioned problems, according to the features of web services, we propose a policy-based adaptive web services security framework (PAWSSF) in terms of its authorization architecture and policy formulation. By defining high-level rules, it can control and adjust web services behaviors without terminating their execution and without modifying the related execution mechanism. Moreover, this framework includes context[6] of user and web services that makes access control adapt to the changed environment.

The rest of this paper is organized as follows. Sect 2 introduces the logical architecture of PAWSSF. The management of context information is given in Sect 3. Sect 4 presents the policy description language ReiT and a mixed reasoning mechanism of rules and ontology. Sect 5 introduces a policy aware BDI agent framework. The prototype system of PAWSSF is implemented in Sect 6. Sect 7 is a case study to describe the usage of ReiT to define web services access control policy. Sect 8 is related work. Sect 9 is the conclusion and future work.

II. A POLICY-BASED ADAPTIVE WEB SERVICES SECURITY FRAMEWORK

In this section we describe how web services access control decisions can be achieved in the PAWSSF which is developed based on the IETF[7] policy framework. The architecture of PAWSSF is shown in Figure 1.

The main modules of PAWSSF are as follows:

① Context management: The user context and web services context are stored respectively in user context repository and web services context repository.

② Policy management: Web services policies described by ReiT are stored in policy repository. Administrator can modify the policies using the policy admin tool. The policies are evaluated by the policy reasoner which is based on the rules and ontology[8].

③ Agent management: There are three types of agents[9]: (ⅰ) User agent, (ⅱ) Web service agent: It acts

Project number: 60903130, BK2007074, BK2009698, BK2009699.
Corresponding author: Bin Li, Yangzhou University, Yangzhou, China. Email: lb@yzu.edu.cn.

doi:10.4304/jsw.6.12.2456-2463

as the Policy Enforcement Point (PEP), (iii) Policy decision agent: It acts as the Policy Decision Point (PDP). With the context of user and web service, the policy decision agent can call the ReiT reasoner to evaluate the web service access policy which are fetched from a policy repository.

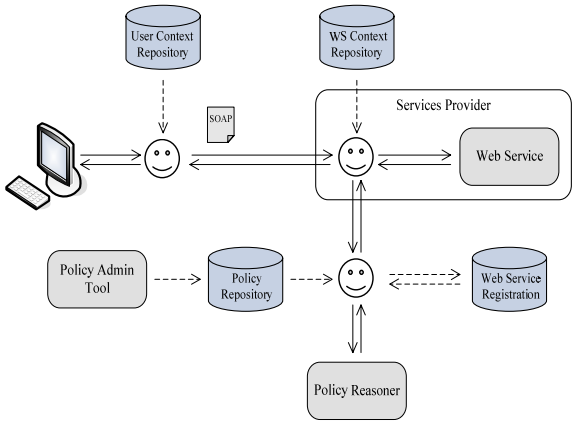


Figure 2. Policy-based adaptive web services security framework

The sequence diagram of successfully invoking the web service is shown in Fig.2.

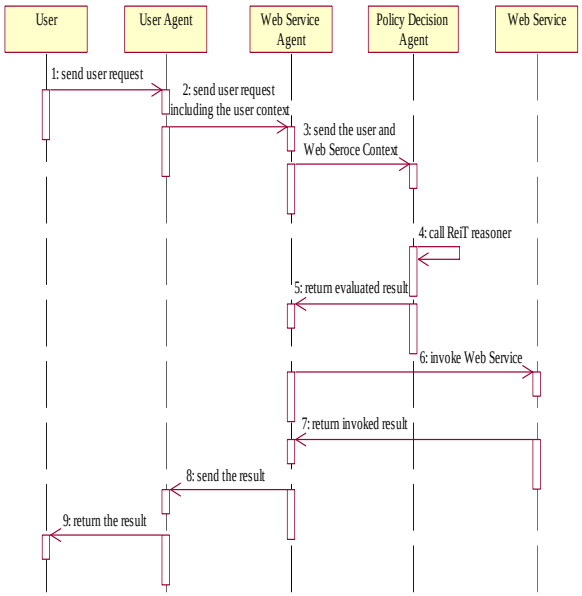


Figure 2. The sequence diagram of successfully invoking the Web Service

III. Context Information Management

Users and web services context is constantly changing in an open environment. To make access control decisions dynamically adapt to the environment change, PAWSSF should sense the context information of users and web services real-time, collect and manage the dynamic information of users and web services.

A. Context information ontology

For access control purposes, there are two types of context in PAWSSF:

① User context: user agent collects associated context which define the characteristics of the user, such as identifier, name, organization, role, etc.

② Web services context: web services have context that can play a part in making access control decisions, and a variety of web service context may be relevant to access control purposes, such as ownership, service taxonomy, state, QoS, etc.

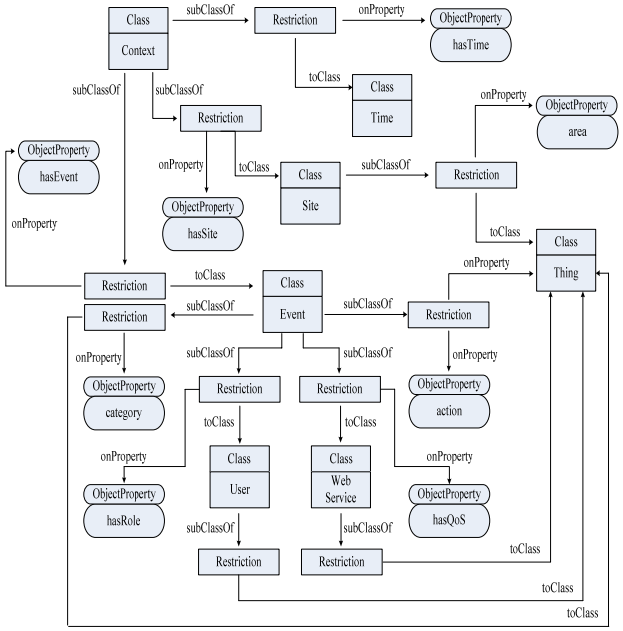


Figure 4. part of Context information ontology

B. Context information encapsulation

In PAWSSF, use agent, web service agent and policy decision agent interaction each to transmit context information. As the transmission process of context information ontology consumes a lot of network resources, the context information ontology is stored centralized and only the context information associated with policy is transmitted. Also, to take into account compatibility with the existing platforms, the context information of users and web services is encapsulated with SOAP message and FIPA ACL message respectively. Use agent and web service agent interact with SOAP, web service agent and policy decision agent interact with FIPA ACL.

① Context information encapsulation with SOAP message

In PAWSSF, context information transmit as a header block of SOAP message, context information header block is composed of several context information blocks, each of which is associated with a sort of context information. Each type of the context formation appears only once. As shown in Fig.4:

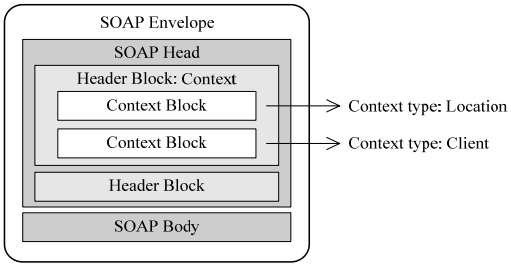


Figure 4. SOAP message with context information

In Fig.4, the context information header block contains two context information blocks, one is corresponding to Location type (location of user currently), and the other is Client type (the software and hardware information of user).

② Context information encapsulation with FIPA ACL message

In PAWSSF, context information stored in the content of FIPA ACL message is the same as SOAP message, the part of context information is composed of several context information blocks, each block associated with one sort of context information, as shown in Fig.5:

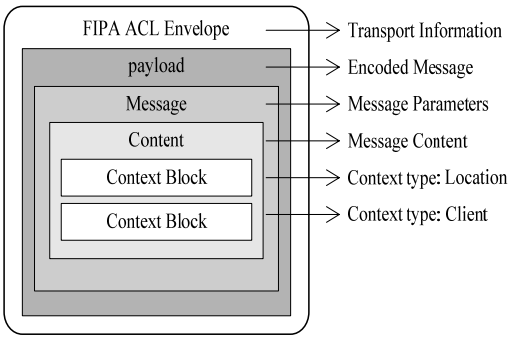


Figure 5. FIPA ACL message with context information

IV. POLICY MANAGEMENT

A. Policy description language ReiT

In PAWSSF, the ReiT, which can be based on any combination of users and web services context, is semantically richer and more expressive. The ReiT policy is based on rules and ontology. It is mainly represented in the form of rules which is based on the Rei[10]. Since the temporal has an important place in open environment, we add the temporal concept which acts as a general element. Moreover the temporal concept is represented by ontology.

Rules component of ReiT

The rules of ReiT mainly include two components:

① Basic Policy: Right, Obligation, Prohibition and Dispensation;

② Interaction Policy: Delegate, Request, Revoke and Cancel.

Basic policy. There are four types of basic policy in ReiT.

① Right Policy: The right policy specifies that the policy subject is allowed to carry out a series of actions to other agents or to the environment. It is described as follows:

Conditions Right(Subject, Action)

Among them, Conditions is an expression. The ReiT policy permits complex conditions to be built from the basic conditions and supports the following operators (and, or, not); Right(Subject, Action) represents that the Subject has the power to carry out the Action.

② Obligation Policy: The obligation policy specifies a series of actions that the policy subject must be carried out. It is described as follows:

Conditions Obligation(Subject, Action)

Among them, Obligation(Subject, Action) represents that the Subject has the obligation to carry out the Action.

③ Prohibition Policy: The prohibition policy specifies a series of actions that the policy subject is not allowed to carry out. It is described as follows:

Conditions Prohibition(Subject, Action)

Among them, Prohibition(Subject, Action) represents that the Subject is not allowed to carry out the Action.

④ Dispensation Policy: The dispensation policy specifies actions that the subject must carry out in the past, but not need to do now. It is described as follows:

Conditions Dispensation(Subject, Action)

Among them, Dispensation(Subject, Action) represents that the Subject doesn't need to carry out the Action any more.

Interaction policy. The ReiT policy language can describe the interactive behaviors. There are four types of policies offered to influence the interaction between the subject and other entities.

① Delegate Policy: A delegation policy allows an entity to give a right to another entity or group of entities. It is described as follows:

Conditions Delegate(Sender, Receiver, Right(Receiver, Action))

Among them, Sender is the sender of the right and the Receiver is the receiver of the right. Delegate(Sender, Receiver, Right(Receiver, Action) represents that the Sender gives its right to Receiver.

② Request Policy: There are two kinds of request policy: a request for an action and a request for a right. It is described as follows:

(ⅰ) request an action

Conditions Request(Sender, Receiver, Obligation(Receiver, Action))

The whole rule represents that the Receiver has the obligation to perform the Action when the request is accepted and the Conditions are true.

(ⅱ) request a right

XConditions Request(Sender, Receiver, Right(Sender, Action))

The meaning of the whole rule is that the Sender is allowed to perform the Action when the request is accepted and the XConditions of Sender are true.

③ Revoke Policy: Revocation policy is the removal of right. It is described as follows:

Conditions Revoke(Sender, Receiver, Prohibition (Receiver, Action))

It represents that the Receiver is prohibited to perform the Action when the Conditions are true.

④ Cancel Policy: An entity can cancel any request sent by the entity itself. It is described as follows:

(ⅰ) Cancel an action

Conditions Cancel(Sender, Receiver, Dispensation(Receiver, Action))

It represents the Receiver is no longer to perform the Action when the Conditions are true.

(ⅱ) Cancel a right

Conditions Cancel(Sender, Receiver, Prohibition(Sender, Action))

It represents that the Sender no longer has the right to perform Action when the Conditions are true.

Conflicts occur if policies overlap in subject, target and action but the policy objects are different. We use the priorities to resolve it. Priorities can be specified between policy rules. For instance, the predication Overrides(A, B) means that A is prior to B, and the conflict between the two rules is resolved at run-time.

Ontology component of ReiT

As we mentioned above, the ReiT policy language is not only the rules to describe the non-structural facts but also the ontology given in figure 3 to describe the structural temporal facts and the vocabulary in ontology can be appeared in rules.

To evaluate the policy, we propose a mixed reasoning framework based on the rules and ontology in the following section.

B. Mixer reasoning framework based on the rules and ontology

We present a pragmatic approach using the Jess and Racer to reason the ReiT policy. The mixer reasoning framework shown in Figure 6 consists of the following steps:

Step 1 The ontology instances involved in ReiT are mapped to Jess facts after reasoning by Racer;

Step 2 The rules in ReiT are mapped to Jess rules and facts;

Step 3 The new Jess facts inferred from the Jess rules and initial facts;

Step 4 Map the new Jess facts to rules, evaluate the policy and add new rules to ReiT knowledge.

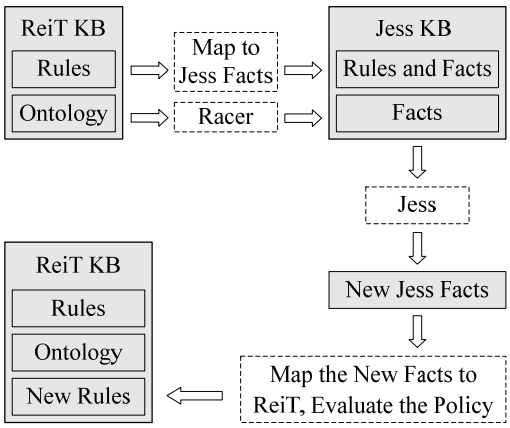


Figure 6. Mixer reasoning framework based on the rules and ontology

V. AGENT MANAGEMENT IN PAWSSF

The policy-based web service security framework mainly includes three types Agent: User Agent, Web Service Agent, and Policy Agent:

① User Agent, the function of which includes surrogating users to submit web service request call; getting users' context information and storing that in the user context database; encapsulating users' context information into SOAP message; sending the SOAP message with users' context information to Web Service Agent; receiving the result of Web Service Agent.

② Web Services Agent, the function of which includes receiving the service call requests from User Agent; parsing SOAP messages sent by User Agent and extracting users' context information; encapsulating the context information of users and web services into FIPA ACL message; sending the context information of users and web services to Policy Decision Agent; receiving the assessment result from Policy Decision Agent; calling the corresponding web service.

③ Policy Decision Agent, the function of which includes receiving the FIPA ACL message with context information from Web Service Agent; extracting the context information of user and web service; querying the web service's WSDL document from UDDI; looking up the access control policy of the web service from policy database; calling ReiT policy inference engine to assess the access policy; returning the assessment result to Web Service Agent.

In the PAWSSF, the behaviors of web services, which are represented by web services agent, are acted according to the policies. In a sense, the policy decision agent is a policy aware agent.

A. Construction of policy aware BDI Agent

We adopt the BDI agent model[11] as the basis of our agent making internal decision. Our basic approach is to use right policies to define what actions an agent is permitted or forbidden to do. The agent is responsible for ensuring that these constraints interact with its beliefs, desires and intentions at the reasoning stage, before acting towards its goals.

We propose a policy aware BDI agent whose state is viewed as consisting of mental components such as beliefs, desires, intentions, capabilities (modeled as right) and commitments (modeled as obligation). With the above notions, we propose a policy aware BDI agent.

The construction of the policy aware BDI agent consists of the five steps, as shown in the Fig.7.

Step 1 We use the rules in ReiT policy file with XML+OWL to describe the agent behaviors;

Step 2 We parse the policy file including the syntax analysis, XML parser and interpreter. The agent is responsible for obeying these policies;

Step 3 After evaluating the applicability of the policies, the ReiT rules are mapped to Jess rules;

Step 4 The internal BDI rules are translated to Jess rules; After merging the ReiT rules and BDI rules we send these rules to ReiT reasoner;

Step 5 With the reasoning result, the agent can act according to its Beliefs and Desires, constrained by the Obligations and Rights.

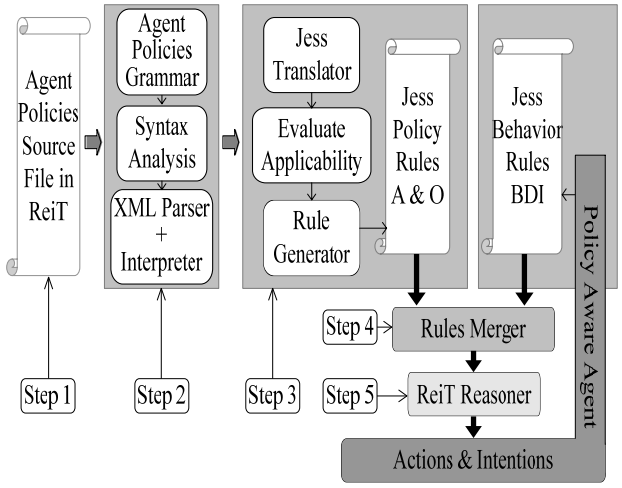


Figure 7. The construction of the policy aware BDI agent

VI. The Implementation Of PAWSSF Prototype System

A. The Implementation of prototype system classes

The classes of prototype system include: context management class, policy management class, web services class and Agent classes, such as user agent, web service agent, policy decision agent.

① context management class, which can save, get, update and delete the context information of web services, and encapsulate context information into SOAP message and FIPA ACL messages, etc.

② policy management class, which can save, get, update and delete policies, and asses ReiT policy, etc.

③ web services class, which can query the description documents of web services from UDDI, receive requests through SOAP message, and return request results, etc.

④ User agent class, which can surrogate users to submit web services requests call, get user context information and store that in the user context database, encapsulate user context information into SOAP message, send the SOAP message with user context information to web service agent, and receive the results of web service agent, etc.

⑤ Web services agent class, which can receive the service call requests from user agent, parse SOAP messages sent by user agent and extract user context information, encapsulate the context information of users and web services into FIPA ACL message, send the context information of users and web services to policy decision agent, receive the assessment result from policy decision agent, and call the corresponding web service, etc.

⑥ Policy decision agent class, which can receive the FIPA ACL message with context information from web services agent, extract the context information of user and web service, query the web services WSDL document from UDDI, look up the access control policy of the web services from policy database, call ReiT policy inference engine to assess the access policy, and returning the assessment result to web services agent, etc.

The classes diagram of prototype system is shown in Fig.8.

The system is implemented on Java EE platform[12] and JADE Agent platform[13].

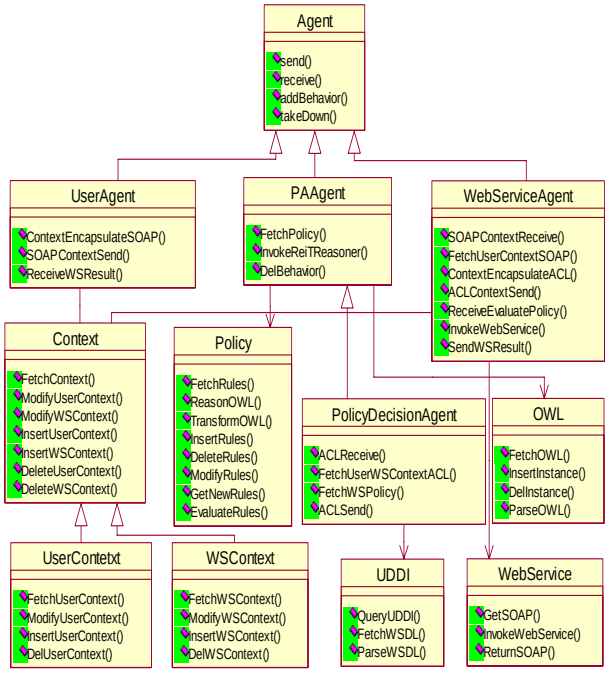


Figure 8. Classes Diagram of prototype system

B. Prototype System Performance Analysis

For web services with different sized tasks, consider the web services execution time with and without access control respectively, the run time of web services working with multi-jobs is shown in Fig.9.

For the same web services, consider the relationship between the number of concurrent users and the response time of system, the web service execution time with and without access control are shown in Fig.10.

The experimental results show that the web services security framework is reasonable and feasible.

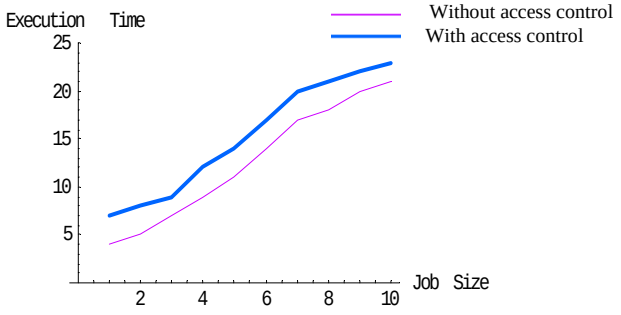


Figure 9. System response time analysis diagram

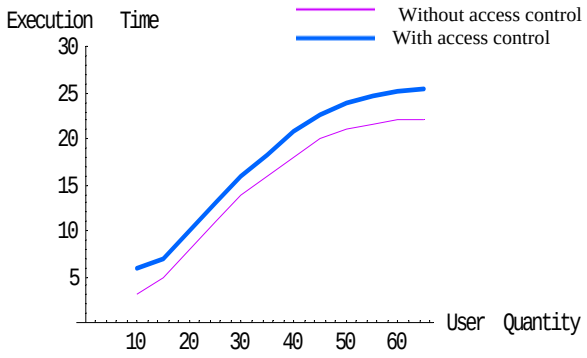


Figure 10. System Reliability Analysis Diagram

VII. A Case Study

In this section, we use a case study to introduce how to use ReiT policy language to describe the access control of the web service.

① Access control policy of web service

For a given weather forecast web service webservice1, the access control policy exists in the policy repository as follows:

(ⅰ) User must login from the given IP (202.195.48.53) to access the webservice1;

(ⅱ) User can access the webservice1 during the interval (01:00:00 - 23:00:00);

(ⅲ) User can access the webservice1 when the load of the webservice1 is less than the max load (10).

The policy of webservice1 mentioned above can be described as follows:

(LoginIP(user, uip) ∧ SameIP(aip, uip) ∧ before(t, t1) ∧ after(t, t2) ∧ WS(ws) ∧ Less(curload, maxload)) Right(user, ws)

Among them, LoginIP(user, uip) represents the user logins from the IP of uip; SameIP(aip, uip) means that address aip and uip are the same IP, and the value of aip is given "202.195.48.53"; before(t, t1) means that the instant t is before the instant t1 , after(t, t2) represents that the instant t is after the instant t2; WS(ws) means that ws is a Web Service; Less(curload, maxload) represents that the current load of the webservice1 is less than maxload, and the value of maxload is given "10"; Right(user, ws) means that user can access the ws.

Moreover, Assuming that Jimmy has the right to delegate, he delegates to Bob the right to access the webservice1 as long as he is working on the same project as Jimmy.

(SameProject(sender, receiver) ∧ WS(ws)) Delegate(sender, receiver, Right(receiver, ws))

Among them, SameProject(sender, receiver) means that sender and receiver are working on the same project, Delegate(sender, receiver, Right(receiver, ws)) represents that sender delegates the right of accessing the ws to receiver.

②User submits web service request in PAWSSF platform

Jimmy logins PAWSSF platform, the address of submitting weather forecast is: http://www.webservicex.net/WeatherForecast.asmx,

the address of providing SOAP Action is : http://www.webservicex.net/GetWeatherByZipCode,

the postcode required when query weather is stored in ZipCode.xml. As shown in Fig.11.

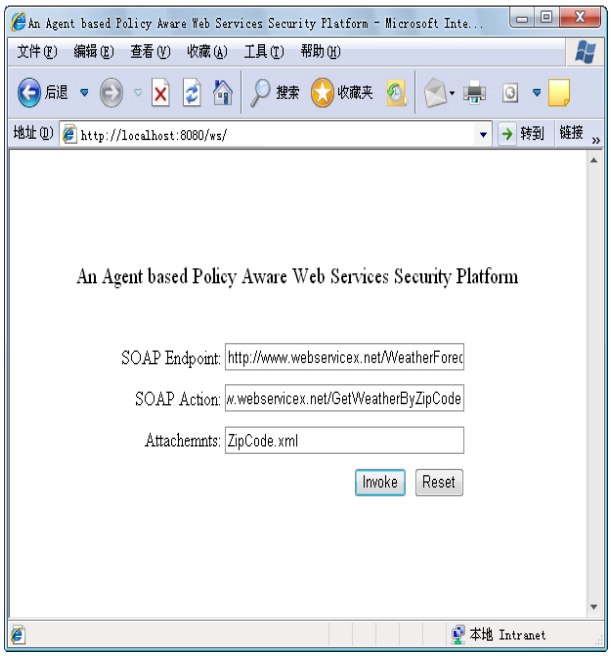


Figure11. User submit the Web Services with PAWSSF

③Agents Construction

After user agent submits request, system creates three Agents, User Agent (useragent), Web Services Agent (wsagent), and Policy Decision Agent (pdagent).

Useragent and wsagent query the context information of user and web services respectively, then pdagent assesses the access control policy of the web services. As shown in Fig.12.

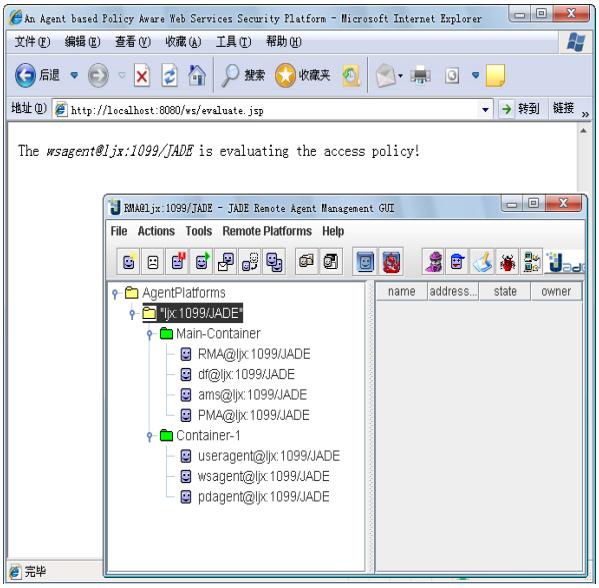


Figure12. Policy Decision Agent construction

④ Context information

Suppose that Jimmy access the webservices1 at time "22:30:00" from the IP of "202.195.48.53". And the webservice1's current load is "7".

⑤ Policy evaluation

With the context of Jimmy, it makes a call to the ReiT policy reasoning engine.

(ⅰ) Ontology reasoning

There are following facts in the temporal ontology: Instant(01:00:00), Instant(23:00:00), Instant(22:30:00), before(01:00:00, 22:30:00) and before(22:30:00, 23:00:00). Racer can infer the new fact "after(23:00:00, 22:30:00)" based on the relation between before and after in the temporal ontology.

(ⅱ) Rules reasoning

After mapping the fact "after(23:00:00, 22:30:00)" to the format that Jess supports, the new fact inferred by Jess is as follows:

Right(user Jimmy) (accws webservice1)

It means that Jimmy has the right to access the webservice1. Additional, the known fact that Bob and Jimmy are working on the same project can be represented as follows:

(assert (SameProject(sender Jimmy) (receiver Bob)))

The new fact can be inferred by Jess is as follows:

Right(user Bob) (accws webservice1)

It means that Bob has the right to access the webservice1. The value of the attribute "Effect" is modified into "Permit" after mapping the fact to ReiT.

⑥ Invoke the web service

As the result of policy evaluated by policy decision agent is "Permit", Web service agent invokes the webservice1 and returns the result to the user agent after the webservices1 invoked successfully.

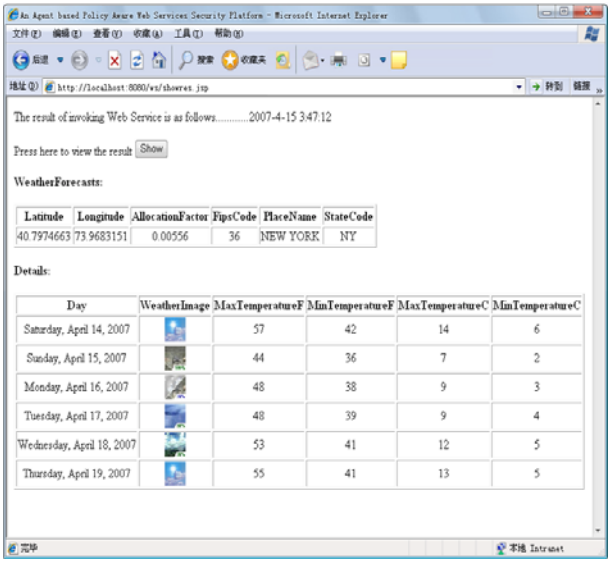


Figure13. the results returned by Web Services Agent

VIII. Related Works

Currently, there are many access control models, such as RBAC, TBAC and PBAC, etc.

RBAC model simplifies the authorization by relating the user's permission to the role of the user. However, it has the drawback of fixed role diffusion that causes increased burden on system management. From the viewpoint of task oriented, TBAC model is constructed for the application rather than the system. PBAC presents the concept to provide action before being authorized. But the behavior verification is difficult in open environment.

There are a lot of languages used to define policy, such as Poder[14], KAoS[15] and Rei, etc.

Ponder uses the object oriented idea to define policy. But the lower abstract levels provided by calling methods limits its ability. KAoS defines policies based on ontology. But it is difficult to define specific policies by using OWL alone. Although the Rei policy is described by OWL-Lite, there are no new conclusions to be drawn.

IX. Conclusion and Future Work

This paper presents a policy-based web services security framework. Innovations in this paper are as follows: ① The PAWSSF has the feature of loosely-coupled as separating the policy decision point from the policy execution point; ② The policy evaluation is based on the context of the user and web service. It is helpful to improve the policy adaptation in the open environment; ③ The ReiT policy based on rules and ontology can express the structural and non-structural knowledge; ④ We propose an effective mixed reasoning framework based on rules and ontology. ⑤ We propose a policy aware BDI agent to authorize the access control of the web services. In future work, we plan to study how to

verify the consistency between policies. Moreover, the distributed detection of the policy conflict is to be studied.

ACKNOWLEDGMENT

This paper is supported by the National Science Foundation of China under Grant No. 60903130, and the Natural Science Foundation of the Jiangsu Province of China under Grant No. BK2007074, BK2009698, BK2009699.

REFERENCES

[1] M.P. Papazoglou, W.J. Heuvel. Service oriented architectures: approaches, technologies and research issues. International Journal on Very Large Data Bases, 2007, 16(3):389-415.
[2] J. Yu, Y.B. Han. Service oriented computing-principle and application. Beijing:Tsinghua university press, 2006.
[3] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman, “Role-based access control models”, Computer, IEEE Press, Feb. 1996, pp. 29(2): 38-47.
[4] S. Oh, S. Park, “Task-role-based access control model”, Information Systems, Elsevier Science Ltd., Sep. 2003, pp. 28(6): 533-562.
[5] M. Kudo, “PBAC: Provision-based access control model”, International Journal of Information Security, Springer Berlin, Feb. 2004, pp. 1(2): 116-130.
[6] J.M. Serrano, J. Serrat; A.Galis, “Ontology-Based Context Information Modelling for Managing Pervasive Applications”, 2006 International Conference on Autonomic and Autonomous Systems(ICAS'06), IEEE Press, USA, Jul. 2006, pp. 47-52
[7] R. Nabhen, E. Jamhour, and C. Maziero, “A Policy Based Framework for Access Control”, Information and Communications Security, Springer Berlin, Sep. 2003, pp. 2003(2836): 47-59.
[8] H. Wache, T. Vogele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hubner, “Ontology-Based Integration of Information-A Survey of Existing Approaches”, 17th International Joint Conference on Artificial Intelligence(IJCAI01), USA, Aug. 2001, pp. 108-117
[9] D. Weyns, G. Vizzari, and D. Keil, et al., Environments for Multi-Agent Systems II, Springer Berlin, The Netherlands, Feb. 2006
[10] L. Kagal, T. Finin, and J. Anupam, “A Policy Language for A Pervasive Computing Environment”, IEEE 4th International Workshop on Policies for Distributed Systems and Networks, IEEE Press, Jun. 2003, pp. 63-74.
[11] J. Buford, G.Jakobson, and L. Lewis, “Extending BDI Multi-Agent Systems with Situation Management”, Information Fusion, 2006. ICIF’06. 9th International Conference, Jul. 2006, pp. 1-7.
[12] K. Mukhar, et al., Beginning Java EE 5: From Novice to Professional, friends of ED, 2005.
[13] Java Agent DEvelopment Framework Homepage, http://jade.cselt.it/, 2009.
[14] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, “The Ponder Policy Specification Language”, Workshop on Policies for Distributed Systems and Networks, UK, Jan. 2001, Springer-Verlag LNCS, pp. 18-39.
[15] G. Tonti, J.M. Bradshaw, R.Jeffers, R. Montanari, N. Suri, and A. Uszok, “Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder”, The SemanticWeb - ISWC 2003, Springer Berlin, Sep. 2003, pp. 419-437.

Bin Li was born in Yangzhou, Jiangsu Province, China, in 1965. He received the Ph.D. degree in computer application technology from Nanjing University of Aeronautics & Astronautics, Jiangsu, China in 2001.

Currently, he is a Professor of Yangzhou University and conducts research in the areas of service oriented computing, multi-agent system and artificial intelligence.

Lingjun Zhao was born in 1987, M. S. candidate. Her main research interests include service computing oriented computing, software agent.

Junwu Zhu was born in Yangzhou, Jiangsu Province, China, in 1972. He received the Ph.D. degree in computer application technology from Nanjing University of Aeronautics & Astronautics, Jiangsu, China in 2008.

Currently, he is a Associate Professor of Yangzhou University and conducts research in the areas of service oriented computing, Ontology and artificial intelligence.

Jun Wu was born in Yangzhou, Jiangsu Province, China, in 1970. He received the Ph.D. degree in computer application technology from Southeast University, Jiangsu, China in 2005.

Currently, he is a Associate Professor of Yangzhou University and conducts research in the areas of service oriented computing, computer network and formal method.

Distributing Query Plan Operators Using Honey-Bees Algorithm

Maryam Kheirkhahzadeh
Islamic Azad university-Ahvaz branch/ Dept. of Computer Engineering, Ahvaz, Iran
Email: kheirkhah@iauahvaz.ac.ir

Mohammad G. Dezfuli
Iran University of Science and Technology, IUST / Dept. of Computer Engineering, Tehran, Iran
Email: mghalambor@iust.ac.ir

***Abstract*—Data Stream Management Systems (DSMSs) has the advantage of lightweight repositories so they can migrate in a network of hosts regarding their stream resources. We developed a framework (including some systems) based on agents to let users have their own customized query engines which can migrate in network to gain better QoS. Our Data Stream Management Agents (DSMAs) can distribute their queries and clone themselves to run different sub-queries on different hosts regarding the position of related sources. One of the main challenges toward this goal is distribution algorithm for queries. In this paper, we proposed a distributor for our agent-based data stream management framework based on honey-bees algorithm. We compare our honey-bees algorithm to other proper heuristics (genetic algorithm and DPSO) through our experimental evaluations.**

***Index Terms*— Stream, DSMS, SDMS, DSMA, Agent**

I. INTRODUCTION

Nowadays, lots of data stream generators are available and accessible through internet (e.g. sensor networks). These stream resources present promising retrievable information for world-wide users (i.e. queriers). What is missing is an infrastructure to interconnect these heterogeneous stream generation domains into a pervasive system that responds to user's queries. The transparency of this infrastructure will allow users to query it as a single unit made of millions of widely distributed ordinary or even advanced complex sensors (e.g. from thermometers to video cameras).

To have a better envision, imagine the following scenario: a biologist wishes to do research on relations between weather conditions and migrations of zebras in Kenya. Instead of travelling and living in Kenya, he can access to sensor network services deployed there and enjoy an online remote surveillance. In addition, the same resources could be queried continuously by Kenya lifeguards to detect migration of zebras toward out of the Sweetwater reserve.

We call our desired system as Pervasive Stream Information Retrieval Network (PSIRN). These five key requirements of a pervasive information retrieval network play an important role in the design of a PSIRN:

- An easy way to interconnect services and users based on available bandwidth-limited networks. First and foremost, we need an infrastructure to interconnect stream producers (i.e. services) and stream consumers (i.e. query executors). It should be robust and reliable despite unreliable low-bandwidth links.
- Finding appropriate services. PSIRN users should be capable of searching and finding services related to their queries.
- Heterogeneity of service providers and service customers. Different service providers/customers may use different customized softwares.
- Data access and privacy. PSIRN is composed of many different administration domains with different access controls and privacy policies. There should be a data access control in PSIRN that satisfies all of the domain administrators.
- Information trading and QoS. Service authors should be allowed to sell their services regarding specific QoS. In fact, an information-trading contract based on a detailed complex QoS metric [1] should be negotiated and signed by two parties.

In this paper we introduce data stream management agents which are responsible for executing a query in a data stream processing infrastructure. Each DSMA is responsible for running a subset of operators of a query plan in the network. We used JADE middleware to support relations between multiple FIPA-compliant agents. This paper is an extended version of our previous work [2] with a better focus on distribution algorithms. In this paper we used a honey-bees algorithm for operator distribution and compared it with genetic algorithm, binary PSO and centralized algorithm. We show that honey-bees algorithm performs better and is faster than other distributor algorithms.

The rest of the paper is organized as follows. First, we cover the related work in Section II. We introduce our new infrastructure in Section III. We focus on three query distribution algorithms in Section IV(honey-bee, DPSO and genetic algorithm) and demonstrate their experimental evaluation in Section V.

doi:10.4304/jsw.6.12.2464-2471

II. RELATED WORK

Aurora* [3] is a distributed version of the stream processing system Aurora [4], which assumes nodes belonging to a common administrative domain. Medusa [5] is an infrastructure supporting the federated operation of several Aurora nodes across administrative boundaries. After all, Borealis [6] is a second-generation distributed stream processing engine that inherits core stream processing functionality from Aurora* and distribution functionality from Medusa. However, none of these systems provides a flexible framework for customized query engines.

The idea of network-aware operator placement for stream processing was considered by Pietzuch et. al. in [7]. They describe a stream-based overlay network (SBON), a layer between a stream-processing system and the physical network that manages operator placement for stream-processing systems. SBON architecture uses a scalable, decentralized, and adaptive optimization technique based on a multi-dimensional metric space called a cost space. A cost space is a d-dimensional metric space where the Euclidean distance between two nodes is an estimate of the cost of routing data between those nodes. We used this distance metric and found it insufficient. They also introduced Network Usage metric and we leveraged their idea to define our Net-cost metric.

One of the areas of interest in optimization problems is swarm intelligence. It is inspired by the social behavior of some insects such as ants and bees. Honey-bees mating optimization (HBMO) is a swarm intelligence optimization algorithm that models the behaviors of bees. Honey-bees algorithms were used to model agent-base systems [8]. We used honey-bees to model our agent based query distributor algorithm. In the experimental evaluation section we show that this algorithm is better than PSO and genetic algorithm on which it is fast and tries to avoid local minima. Ref. [9,10] show independently that genetic algorithms could be elegantly useful to optimize database query plans. Ref. [11] represents a Discrete Particle Swarm Optimization (DPSO) approach for grid job scheduling. We developed a honey-bees query plan distributor and compare it to DPSO and genetic algorithm.

The efficiency and scalability of JADE and its message transportation system for large agent-based software systems has been tested independently in [12,13] and also we found it very efficient in our experiments.

III. THE INFRASTRUCTURE

To overcome low bandwidth and unreliable links, we used mobile query processors. It means we can get a part of query plan to a mobile query processor and then this processing unit can go to an optimal position that is the nearest position to its needed stream sources. On the other hand, mobility is not possible for heavyweight DBMSs because of their high volume data but it is good for lightweight DSMSs (that contain only queries and some data sketches). Thus we need a multi-agent system because each service provider or service customer works on behalf of its owner. We also need a layer to simplify wide access to data stream sources that supports service oriented architecture (SOA), compatible with internet, supports different kinds of DSMSs and supports mobility of DSMAs.

Our PSIRN is based on agent-oriented java-based JADE middleware [14,15]. JADE is a framework to develop multi-agent systems in compliance with FIPA specifications [16]. JADE could be run on heterogeneous platforms from powerful servers to cell phones. It also supports mobile agents and yellow pages (to find service providers) and facilitate message passing between agents.

It has an acceptable performance and scalability even for heavy loads [12,13]. We have made the simplifying assumption that there is only one administration domain and we leave the security, QoS and information trading for future work.

A. The Architecture

Each site runs a middleware whose major components are shown in Fig. 1, 2. The architecture is based on four types of agents:

1) Data Stream Management Agents (DSMAs)
2) Global Service Directory (GSD)
3) Stream Proxies (SPs)
4) Topology Explorers (TEs)

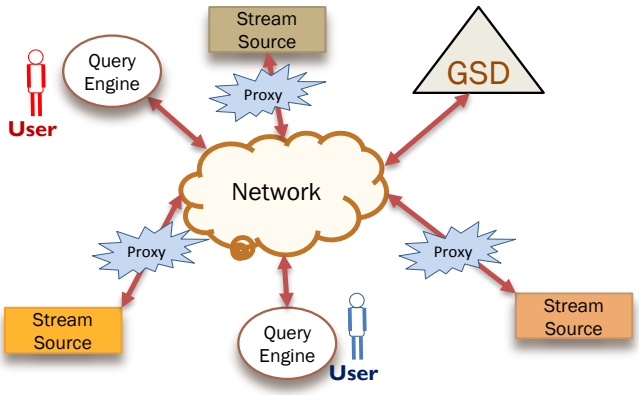


Figure 1. The Topology

DSMAs are agent-based DSMSs. The DSMA is a mobile agent containing a pre-designed and tested DSMS engine.

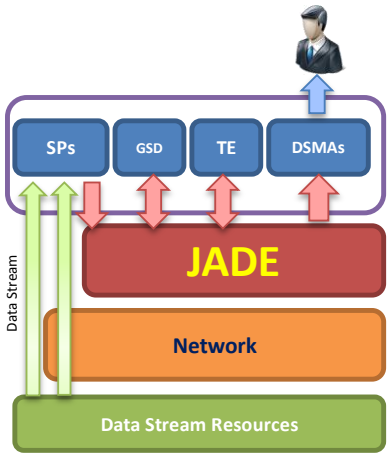


Figure 2. The Architecture

The GSD is an immobile agent that stores information about services. Users can find proper services using GSD. One GSD is sufficient for an administration domain. The SPs are customized interfaces between stream sources and service customers. Each SP is well adapted to a stream source and should register its services in the GSD and provide service for DSMAs in a standard pre-defined manner. In fact, SPs are mobile agents and they travel to the best host regarding their stream sources. There is one immobile TE for each host. TEs compute the network delays between their host and other hosts. TEs help each other to make a Virtual Network Model (VNM) based on peer to peer network delays. TEs update This VNM without any interaction with network layer and it helps other components to look at network layer as a black box. VNM is essentially needed for optimal query distribution in DSMAs.

B. The DSMA Architecture

The DSMA architecture is illustrated in Fig. 3. Some modules like Query Plan Manager, Memory Manager, Scheduler, QoS Manager and AAA (Authorization, Authentication and Admission Control) are similar to their counterparts in DSMSs. The GUI helps user to search and find proper data sources and enter queries. It also displays the results of the query. The JADE Interface Module is an interface between DSMAs, GSD, SPs and TEs. It handles the message passing between mentioned agents. Input ports of DSMA should be connected to JADE interface and output ports could be connected to a file, GUI or JADE interface. Query Decomposer is the most complex and important module that we added to DSMA. Query Decomposer is responsible for decomposing queries using some QoS metrics and network model. It requests Migrator to send different cloned DSMAs to selected hosts. Migrator is responsible for migration of these new generated DSMAs.

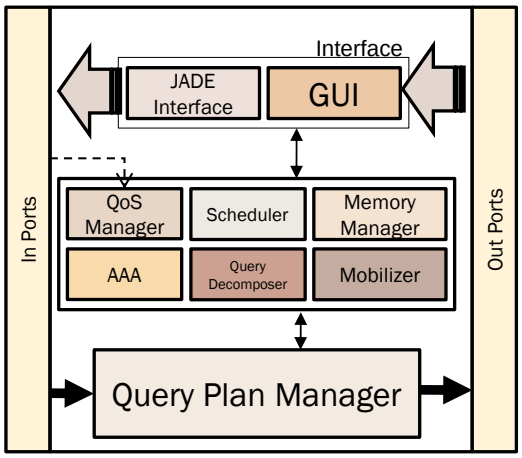


Figure 3. The DSMA Architecture

C. The Connections between Agents

We used Internal Message Transport Protocol (IMTP) of JADE to interconnect agents. We also used XML instead of the JADE's ontological approach for our messages. Agents use yellow pages service of the JADE to find basic services like the GSD.

JADE names each agent with a globally unique identifier. It also can automatically handle the mapping between name of the agent and its physical address. JADE does this mapping using a mapping table in the main container. Other containers use cached mapping table to speed up the mapping process. It helps to prevent any bottleneck in the main container. Agents can communicate directly through their containers and without interference of the main container. Network layer is responsible for low-level message routing.

IV. QUERY DISTRIBUTION

The most important and complex part is how to decompose queries and distribute sub-queries on the network. DSMAs distribute query plans regarding VNM and the locations of producers and customers. Similar cloned DSMAs migrate to selected hosts carrying different parts of the decomposed query. There are two important objective functions to achieve an optimal query distribution: 1) minimizing network usage and 2) minimizing response time of queries.

There are three challenges when facing the query distribution problem: 1) modeling topology of the network (i.e. VNM), 2) decomposing queries into some sub-queries and 3) sub-query placement. The second and third challenges are toward an objective function and are tightly related so they could be covered by a single method (as we did).

A. Virtual Network Model

Our infrastructure lies upon the network layer and treat it as a black box. This helps having an infrastructure for many different interconnected networks and it is necessary for internet infrastructures. Because of the hidden aspects of the underlying network, we need a virtual network model to achieve a near optimal query distribution. The VNM should be creatable without any help from the network layer.

We model network as a complete weighted graph. In this graph, the communication delays between hosts (i.e. vertices) are modeled as weights of the edges (i.e. virtual connection links). This model is proper for different mentioned cost functions (i.e. two objective functions for optimization). In addition, middleware can generate and update this model independent of underlying network layer. The only drawback of VNM is high cost of updating weights of the edges (it has a complexity of $O(n^2)$ where n is the number of hosts). We can improve the cost using minimum spanning tree to bring the complexity down to $O(n)$. Inter-domain distribution is not allowed (because of the policies of administration domains) and sub-queries should be sent to administration domains to be distributed by them locally. Thus, the current updating cost is acceptable for us.

TEs are responsible for generating and updating VNM. They do it using permanent peer-to-peer messaging. DSMAs can get the model from main TE of the administration domain and use it to get a near-optimal decomposition and distribution.

B. Modeling Query Plans

Query plans could be derived from registered continuous queries. They are composed of operators, which perform the actual data processing; queues, which buffer data as it moves between operators; and synopses, to hold state of operators.

Operators are the basic data processing units in a query plan. An operator takes one or more streams as input and produces a stream as output. As in a traditional DBMS, a plan for a query connects a set of operators in a tree: The output of a child operator forms an input of its parent operator, the input streams and relations of the query form the input of the leaf operators, and the output of the root operator forms the output of the entire query. Because of the variety of operator types, we model operators as black boxes with multiple inputs and outputs. For each operator O, D(O) is the processing delay for each input tuple and R(O) is the ratio of output to input (rate of output generation).

C. Query Decomposition and Distribution

Distributing *m* operators among *n* hosts is a NP-hard problem. We developed our distributor based on honey-bees, PSO and genetic algorithm and compared them to each other. We show that honey-bees is the best one. In this section we explain in brief each algorithm and its encoding schemes that we used. Encoding is a mapping from knowledge domain to the solution space where our algorithms can process. The selection of encoding scheme is varied with the design decision and also depends on the problem to be solved. In all algorithms, we model each solution by a $1 \times n$ array. *n* shows the number of operators in the query plan. Each cell *i* of the array contains a host number that we assigned *i*th operator to it.

C.1 discrete PSO

Particle swarm optimization (PSO) is a population based stochastic optimization technique introduced by Kennedy and Eberhart [17]. PSO simulates social behavior of bird flocking. The algorithm is initialized with a population of random solutions. Each solution represents a particle. All particles move based on pbest (personal best) and gbest (global best) in the search space to find the best location. $pbest_i$ is the best location that *i*th particle has experienced so far. gbest stands for global best and is the best location that all particles have experienced so far. Pbest and gbest should be updated after each iteration of the algorithm. The algorithm repeats until a threshold is reached or it finds the optimal solution. In fact, after each iteration, the position of each particle updates with the velocity vector. Velocity vector is calculated based on pbest and gbest.

PSO is applicable in many fields such as function optimization, artificial neural network training, fuzzy system control. There are two versions of PSO algorithms. The basic PSO is suitable to solving continuous problems. The second version named binary PSO is capable of solving discrete problems and introduced by Kennedy and Eberhart [18]. Similar to the work in [18], we used binary PSO to solve our discrete optimization problem because our goal is to assign proper host numbers to the operator set and the search space of our algorithm is discrete instead of continuous (domain of host numbers is a discrete set). We also used DPSO [11] algorithm that is a discrete version of PSO and it has a high performance.

C.1.1 Encoding Scheme

Izakian et. al. proposed an efficient discrete PSO algorithm with a direct representation named DPSO [11]. We can use two representations for a discrete PSO problem to encode each solution: 1) indirect and 2) direct. In indirect representation, each particle must be modeled by a two dimensional vector $h \times n$. *h* shows the number of hosts and *n* is the number of operators. If the operator *i* is assigned to the host number *j*, the proper cell should be one, otherwise zero. That means cells show what operator has assigned to which host by the one or zero value. However, direct representation uses a $1 \times n$ vector for each particle which *n* is the number of operators and each cell has an integer value shows the number of host that assigned to that operator. We used efficient direct representation DPSO algorithm. We chose global best instead of local best to speed up the algorithm. Each particle modeled as a $1 \times n$ array. Gbest and pbest are also $1 \times n$ arrays. We modeled velocity vector, with a $p \times h \times n$ array. The *p* parameter is the number of particles, *h* is the number of hosts and *n* is the number of operators in the query plan.

C.1.2 distribution algorithm

Fig. 4 shows the discrete PSO distribution algorithm that we used. PSO algorithm first generates initial random particles and then assigns each particle to its pbest. After that, it assigns the best pbest to the gbest. In the *while* loop in step 2, PSO calculates the fitness value of each particle and then updates pbest and gbest. Thus, PSO can calculate velocity vector of each particle by updated gbest and pbest. Note that our velocity updating function is like DPSO[11]. Finally, new particles can be generated using new velocity vectors.

```
1.      Initiate random particles and assign
        each particle to its pbest.
        Find initial global best.
2.      While have enough time
2.1.      Calculate fitness of particles
2.2.      For each particle Update pbest
2.3.      Update gbest
2.4.      For each particle do these steps
          with DPSO algorithm
2.4.1        Update its velocity vector
2.4.2        Update its position
3.      Return gbest particle as solution
```

Figure 4. Discrete PSO based distribution algorithm

C.2 Honey- bees algorithm

The idea of honey-bees algorithm is to simulate the behavior of honey bees. There is a queen in the honey bees' colony. Queen starts a mating flight with an initial speed and energy and mates with drones. After each

mating flight, the speed and energy of queen will be decreased. Queen returns home when its speed or energy is near zero or queen's sperm repository is full.

Honey-bees algorithm is composed of genetic, simulated annealing and local search algorithms. It uses simulated annealing for selecting best chromosomes in the selection phase of genetic algorithm. Using simulated annealing in the honey-bees algorithm reinforces it to escape from local minima. It is an important characteristic of simulated annealing algorithm because it also accepts worse solutions with a probability. We show in our experimental evaluation that honey-bees does not get trapped in local optima. Other benefit of using simulated annealing in the selection phase is that only the strongest drones will be selected so the number of times the fitness function invoked will be decreased. The probability of selecting each drone in honey-bees algorithm determined by (1).

$$P(Q,D) = e^{-ABS(\frac{f(Q)-f(D)}{speed})} \quad (1)$$

In the above formula, *f* is a fitness function. *f*(Q) is fitness value of queen and *f*(D) is fitness value of drone. The speed of queen decreases after each iteration of the algorithm so the selection probability for a drone in the initial steps is more than next steps. The difference between *f* (Q), and *f*(D) is another effective factor. In fact, the smaller the difference, the more the selection probability.

C.2.1 Encoding scheme

We modeled each drone and each brood by a $1 \times n$ array. We also defined queen as a $1 \times n$ array.

C.2.2 Distribution algorithm

Fig. 5 shows our honey-bees distribution algorithm. At first, initial population generator function gets the hive size (the number of initial drones) as the input parameter. and generates random initial drones. It also assigns the best one to the queen. The number of queens may be more than one. However, we decided to have one queen in our algorithm.

Ref. [19,20], used random initial values for the initial speed and initial energy of the queen, in step 2. We set the initial speed value to $5e^{10}$ like [21], because it leads to generate better results in our experiments. We used random initial value for energy. While energy is more than zero, we select a drone and then based on its selection probability, which is computable using (1), it may be selected and added to the queen's repository for making a brood. After selection step, we should crossover each selected drone with the queen to make a new brood. Note that queen does not change. The result of our crossover algorithm is an $1 \times n$ array as a new brood. Our crossover algorithm, selects randomly p_1 percent of operators from selected drone and replaces their host numbers with the proper host numbers of the queen. At the next step, mutation function works on the new brood. The role of mutation is to keep the diversity of population. We define two kinds of mutations. First mutation algorithm chooses two random operators from the new brood and swaps their host numbers with each other. Our second mutation algorithm chooses a random operator from the new brood and simply assigns another random host number to it. We named our first mutation function *Mutation1* and the second one, *Mutation2*. We tested two types of our mutation functions and found out that *Mutation2* is more efficient for our honey-bees algorithm and we used it. However you can see in the next section that *Mutation1* is more suitable for our genetic based distribution algorithm.

In step 2.5, if a new generated brood exists that has a better fitness than queen, queen must be replaced with it. Other broods should be saved for the next generation.

In the final step, We kill and remove old drones and make new ones for next generation. Our new generation contains the new broods generated in current iteration plus new randomly generated drones like [19].

```
1.      Generate initial random drones(Hivesize)
        Assign the best drone to the Queen
2.      While have enough time
2.1.      Speed=5e^10
2.2.      Energy=rand[0.5,1];
2.3.      While energy>0&& Queen's Repository
          isn't full
2.3.1.      Select next drone and calculate Δ
            (Δ(f)=fitness(Queen)-fitness(drone))
2.3.2.       Generate r=rand(0,1);
2.3.3.      If(exp(-|Δ(f)/speed|)>r)
              Add current drone to Queen's
              Repository
              Update speed and energy
2.4.        For each drone in Queen's Repository
2.4.1.        crossover drone with the Queen
              with p1%relocation to make a brood
2.4.2.        use Mutation1 for new brood.
2.5.        For each brood do
              If fitness(brood)>fitness(Queen)
                        Queen=new brood
              else
                        Save the new brood
2.6.        Generate new drones by new broods
            plus randomly generated drones
3.      Return queen as solution
```

Figure 5. Honey-bees distribution algorithm

C.3 Genetic algorithm

Genetic algorithms are stochastic search methods based on natural biological evolution and also they are in class of global search methods. J. H. Holland in [22] did much work to develop genetic algorithms. They have been applied to a wide range of optimization problems, including well-known NP-complete and NP-hard problems, scheduling and routing, configuration, and query optimization [9,10].

In genetic algorithms, each solution is modeled as a chromosome and a collection of these chromosomes is called a population. They start with a population that usually generated randomly or may be heuristically. In each iteration, some of the best chromosomes would be selected according to their fitness values. Genetic algorithms have two operators: crossover and mutation.

After the selection phase, the crossover operator tries to make better chromosomes out of selected ones. The goal of mutation operator is to increasing the diversity by applying random changes into the chromosomes.

C.3.1 Encoding Scheme

Encoding scheme will affect the selection of genetic operators. Improper encoding scheme will produce infeasible chromosomes generated by genetic operators. Usually, chromosomes are represented as fixed length strings coded with a binary character set. Other types of encoding schemes include real number representations and permutation representations. We model each chromosome by a $1 \times n$ array. n is the number of operators in query plan and each cell shows the related host that operator assigned to it.

C.3.2 distribution algorithm

Crossover is the most important operator in genetic algorithms. It takes valuable information from both parent chromosomes and then combines them to find a highly fit chromosome. Our crossover algorithm simply selects p_3 percent of operators randomly from a chromosome and then changes their hosts with the hosts in another randomly selected chromosome.

We used *Mutation1* for our mutation function in genetic algorithm. As we explained in section C.2.2, it chooses two random operators from current chromosome and simply swap their host numbers to each other. We tested two types of mutation and found out that this type of mutation is more efficient for our genetic based distribution algorithm. Fig. 6, shows our genetic based distribution algorithm.

```
1.    InitRandomChromosomes (p1 samples)
2.    While have enough time
2.1.    Calculate fitness of chromosomes
2.2.    Sort chromosomes regarding their
        fitness
2.3.    Select p2 number of the best
        chromosomes
2.4.    CrossOver each chromosome with a
        random chromoshome with
        p3% relocations
2.5.    Mutate each chromosome with
        probability of p4 (Mutation1)
3.    Return chromosome with best fitness as
      solution
```

Figure 6. Genetic-based distribution algorithm

C.4 fitness Function

To evaluate our results we need to determine one or more fitness functions. We used two fitness functions: ART-Cost and Net_Cost. Fitness function gives each solution a fitness value which is a judgment of its surviving capability. Choosing and formulating an appropriate fitness function is crucial in obtaining efficient solution for those problems solved by the algorithms.

ART-Cost is the ratio of the average response time to the ideal response time. We compute the ideal response time regarding two conditions: 1) operators work concurrently with maximum delay, and 2) no delay for interconnecting links. For example, ideal response time for query plan in Fig. 7, is equal to 22.

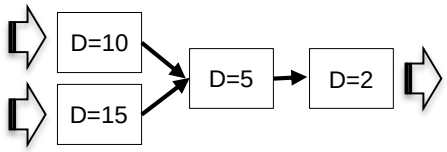


Figure 7. An example for query plan

Net-Cost is the bandwidth-delay product of the query. Net-Cost captures the idea that the longer the data stays in the network, the more likely it is to traverse nodes and links that could be used for other queries [7]. The Net-Cost for a query q, is the amount of data that is in-transit for q at a given instant and calculates by (2).

$$u(q) = \sum_{l \in L} B(l)L(l) \qquad (2)$$

Where L is the set of links used by q, B(l) is the bandwidth of link l, and L(l) is the latency. We can compute fitness function based on these two cost models using approximating (determined statistically) network and query parameters.

V. EXPERIMENTAL EVALUATION

In this section, experiments are conducted to testify the advantages of our proposed approach in terms of Net-cost and ART-cost metrics. All algorithms in the experiments are implemented with java language. We used two simulation-based scenarios to evaluate our distribution algorithm. We compared honey-bees, DPSO and genetic algorithm with each other and with centralized algorithm for each scenario. The centralized method reveals the effect of distribution.

The first scenario is about using high fan-in query plans [23]. We used low fan-in query plans (just like binary trees) for second scenario.

First we show how the fitness value of each distribution algorithm changes over time and also you can see in Fig. 8, that honey-bees algorithm does not get trapped in local minima.

For each operator O, D(O) and R(O) are both equal to 0.5. For honey-bees algorithm, we defined one queen. We selected the initial value of $5e^{10}$ for speed and a random value in range (0.5,1) for energy. Decreasing rate for energy and speed is 0.9. To select the size of queen's repository, we tested lower values than 4 and greater values than 4 and we found that 4 is the best one. Also hive size is set to 20. We set the crossover probability p_1 to 60. In the PSO algorithm, we set $c_1=c_2=2$, $r_1=r_2=1$, $v_{max} = 4$. In addition, in the genetic algorithm $p_1=100$, $p_2= 50$, $p_3=0.6$ and p_4 is equal to 0.3. The data sources and users are hosted by random hosts.

A. Relative changes in values of fitness over time

We executed each algorithm 100 times. Fig. 8, represents changes in average fitness value over time interval (0,40). The Fitness function is Net_cost in high fan-in scenario. You can see that PSO and genetic algorithm get trapped in local minima and fail to reach the goal but honey-bees algorithm performs well because of the benefit of using simulated annealing in its selection phase. In fact, simulated annealing tries to avoid local minima. In addition, Fig. 8, represents that honey-bees algorithm is considerably faster than genetic and PSO algorithms. The selection phase of honey-bees algorithm speeds up it because of selecting strongest drones to mate with queen. Therefore, the number of fitness function, crossover and mutation invocation will be limited to the size of queen's repository.

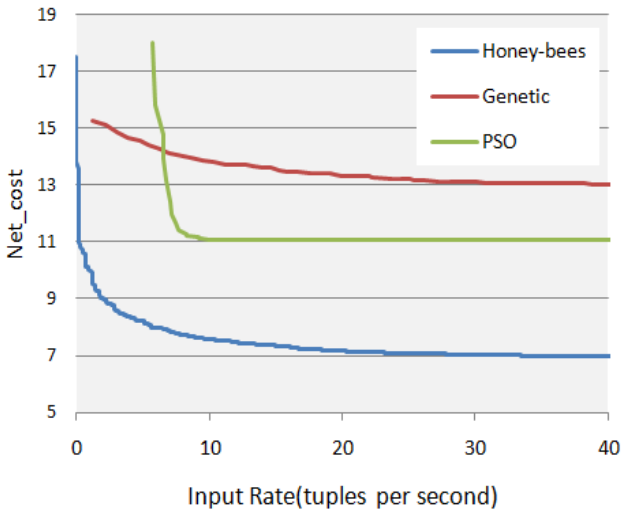


Figure 8. The values of fitness at the first 40milisecond

B. First Scenario: High Fan-in Query Plans

In this scenario, the network model composed of 400 randomly placed hosts. The query plan is a tree with depth of 3 and fan-in of 7 (i.e. each operator has 7 inputs and 1 output). So each query plan has 393 connected operators.

The chart in Fig. 9, shows how Net-cost value changes almost linearly as the input stream rate in the system is increased. In this experiment, we limit the execution time of distributors to 0.8 minute. The effect of changing R and D parameters is just like input rate so we ignore it. Our query distributors are better than centralized algorithm. Honey-bees algorithm is the best distributor and it has a lower fitness value in each step than other algorithms. The results for PSO and Genetic algorithm are almost like each other.

The obtained results in Fig. 11, show how better our honey-bees distributor is in ART-cost metric in respect to the other distributors. Centralized algorithm has a bit lower fitness than honey-bees algorithm. It is trivial because in the centralized algorithm all operators are assigned to one host so the cost of network delay and bandwidth will be decreased. Note that the values of R and input rates are not effective in ART-Cost metric.

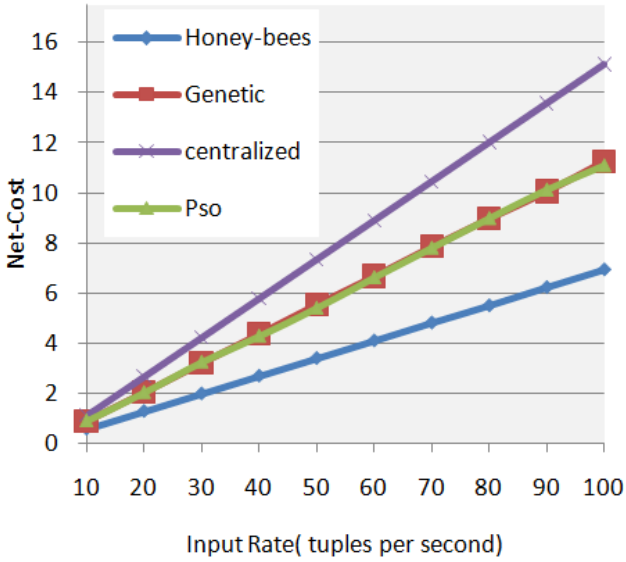


Figure 9. Net-cost for high fan-in queries

C. Second Scenario: Low Fan-in Query Plans

In this scenario, we used usual low fan-in query plans with depth of 2 and fan-in equals to 2 (i.e. each operator has 2 inputs and 1 output). So each query plan has 2 connected operators. Other parameters are just like the previous scenario.

The charts in Fig. 10, 12, show that all algorithms almost operate like each other. Using query plan distributors in low fan-in scenario does not lead to better Net_cost. Hence, distribution does not play an important role in low fan-in scenario. In high fan-in scenario, the centralized algorithm no longer performs well.

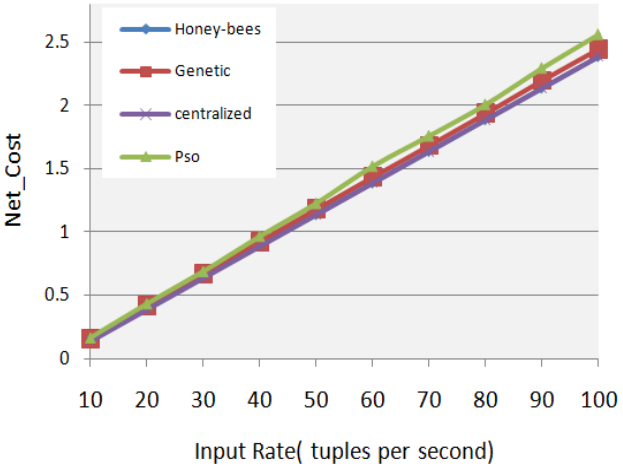


Figure 10. Net-cost for low fan-in queries

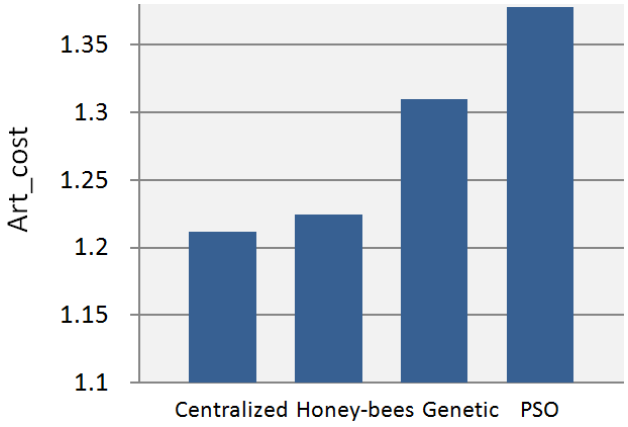


Figure 11. ART-cost for high fan-in queries

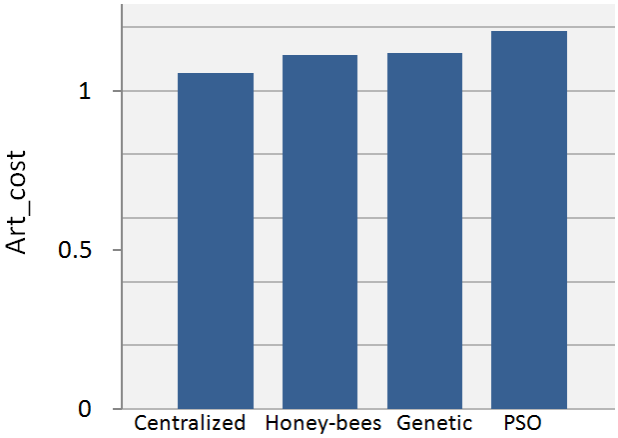


Figure 12. ART-cost for low fan-in queries

VI. CONCLUSION

We introduced a new framework for data stream query processing which is based on an Agent-based middleware, JADE, and allow users to have agent-based query engines which are capable of decomposing their queries to sub-queries and clone themselves to distribute query processing. This idea helps us to process each sub-query near its stream sources and leads to less net costs.

The most challenging and complex module in our systems was query distributor. We implemented our query distributor system with honey-bees algorithm and compared it to DPSO and genetic algorithm. DPSO is the binary version of PSO and suitable for discrete domains. In the experimental evaluation section we showed that our honey-bees based query distributor algorithm is more efficient than two other distributors (esp. for high fan-in queries) and it increases the network utilization.

REFERENCES

[1] M. Ghalambor, A.A. Safaeei, and M.A. Azgomi, "DSMS scheduling regarding complex QoS metrics", IEEE/ACS International Conference on Computer Systems and Applications (AICCSA), 10-13 May 2009.

[2] M. Kheirkhahzadeh and M. G. Dezfuli, "Agent-based pervasive stream information retrieval", In proc. Of the IASTED Int. Conf. on Artificial Applications, AIA 2011, pp.171-176, Innsbruck, Austria, Feb 14-16, 2011.

[3] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, et al., "Scalable distributed stream processing", In Proc. of the 1st Biennial Conference on Innovative Data Systems Research (CIDR), CA, January, 2003.

[4] D.J. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, et al., "Aurora: a new model and architecture for data stream management", The VLDB Journal The International Journal on Very Large Data Bases, Vol. 12, No. 2, pp. 120-139, 2003.

[5] M. Balazinska, H. Balakrishnan, and M. Stonebraker, "Load management and high availability in the Medusa distributed stream processing system", In Proceedings of the ACM SIGMOD international conference on Management of data, NY, USA, 2004.

[6] D. J. Abadi, Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, et al., "The design of the Borealis stream processing engine", In Proc. of the 2nd Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, CA, January, 2005.

[7] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, et al., "Network-aware operator placement for stream-processing systems", In Proc. of ICDE, April, 2006.

[8] A. Pérez-Uribe and B. Hirsbrunner,"Learning and foraging in robot-bees", in Meyer, Berthoz, Floreano, Roitblat andWilson (eds.)', SAB2000 Proceedings Supplement Book, Intermit. Soc. For Adaptive Behavior, Honolulu, Hawaii, pp. 185–194.

[9] M. Stillger, and M. Spiliopoulou, "Genetic programming in database query optimization", In Proc. of the Genetic Programming Conference, pp. 388-393, July 1996.

[10] K. Bennett, M.C. Ferris, and Y.E. Ioannidis, "A genetic algorithm for database query optimization", In Proc. of the 4th International Conference on Genetic Algorithms, 400-407, 1991.

[11] H. Izakian , B. Tork Ladani, A. Abraham, V. Snasel, "A discrete particle swarm optimization approach for grid job scheduling". International Journal of Innovative Computing, Information and Control vol .6, No. 9, September 2010.

[12] E. Cortese, F. Quarta, G. Vitaglione, and P. Vrba, "Scalability and performance of JADE message transport system", Journal of Analysis of Suitability for Holonic Manufacturing Systems, Vol. 3, No. 3, pp. 52-65, 2002.

[13] K. Chmiel, "Efficiency of JADE agent platform", Scientific Programming Journal, Vol. 13, No. 2, pp. 159-172, 2005.

[14] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE--a FIPA-compliant agent framework", Proceedings of PAAM, Vol. 99, pp. 97-108, 1999.

[15] JADE Official Website, http://jade.tilab.com.

[16] P.D. O'Brien, and R.C. Nicol, "FIPA — towards a standard for software agents", BT Technology Journal, Vol. 16, No. 3, pp. 51-59, Jul 1998.

[17] J. Kennedy and R. C. Eberhart, "Particle swarm optimization", Proc. of the IEEE International Conference on Neural Networks, pp.1942-1948, 1995.

[18] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm", IEEE International Conference on Systems, Man, and Cybernetics, Orlando, FL, vol.5, pp.4104-4108, 1997.

[19] N. R. Sabar, M. Ayob, G .Kendall,"Solving Examination Timetabling Problems using Honey-bee Mating Optimization (ETP-HBMO)", (MISTA 2009), August2009.

[20] O. B. Haddad, A. Afshar and M. A. Marino, "Honey-Bees Mating Optimization (HBMO) Algorithm:A New Heuristic Approach for Water Resources Optimization", Water Resources Management (2006) 20: 661–680,DOI: 10.1007/s11269-005-9001-3.

[21] O. B.Haddad, M. Mirmomeni,M. Z. Mehrizi,M.A.Mariño ,"Finding the shortest path with honey-bee mating optimization algorithm in project management problems with constrained/unconstrained resources", Published in Journal of Computational Optimization and Applications, Volume 47, Issue 1, September 2010.

[22] J. H. Holland. "Adaptation in natural and artificial systems", Ann Arbor:University of Michigan Press, 1975.

[23] M.J. Franklin, S.R. Jeffery, S. Krishnamurthy, F. Reiss, S. Rizvi, et al., "Design considerations for high fan-in systems: the HiFi approach", In Proc. Of the CIDR Conf., Jan. 2005.

A Comprehensive Optimization Model Based on Time and Cost Constraints for Resource Selection in Data Grid

Qu Ming-Cheng
School of Computer Science and Technology, Harbin Institute of Technology, Harbin, 150001, China;
Email: qumingcheng@126.com

WU Xiang-hu, YANG Xiao-zong
School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China;
Email: Wuxianghu@hit.edu.cn;yxzong@hit.edu.cn;zuodc@hit.edu.cn

***Abstract*—Parallel data transmission based on multi-copy can enhance transmission speed and ensure the QoS of data grid greatly. The status of network and replica node, the distance of replica node, the time and bandwidth of service requester will directly affect service cost. How to take the above factors into account, so as to provide basis for node selection and bandwidth allocation, guarantee the time constraint of service requestor and optimize service cost is a key problem need to be solved urgently. Based on '0-1' integer programming and linear programming methods, respectively, a minimum transfer time model, minimum cost model and comprehensive optimization model are proposed to solve the above problem. Simulative experiments show that the models are correct and effective.**

***Index Terms*—resource selection optimization model, transmission time constraint, parallel data transmission, data grid, QoS**

I. INTRODUCTION

Reliable and fast data transfer is a basis to guarantee the QoS of data grid [1]. It can greatly improve transfer speed, decrease network traffic and guarantee QoS of grid by deploying data replicas in hot area [2, 3]. GridFTP provides a striped data transmission mode [4-5]. All kinds of parallel transmission algorithms based on GridFTP, by downloading different data block from different replicas concurrently, improve download speed further[6,7].

Chao-Tung Yang [6-8] proposed a parallel transmission algorithm and a replica node selection model, the model considers the parameters of network bandwidth, status of CPU and I/O, and can output the service node set to achieve the objective of minimum transmission time. In order to improve the performance of parallel computing, Dafei Yin [9] proposed a simple replica node selection model, considering data redundancy and parallelism, attempt to establish a compromise between the two parameters. Similarly, Gaurav [10] put forward a linear optimization model to minimize transmission time. Husni [11] also adopted some strategies to select the replica node, and simply guarantee the minimum transmission time, so that the job can be finished quickly.

Related researches about parallel transmission based on multiple copies more focus on maximizing the data transmission speed. However, these studies have certain shortcomings: (1) not all of the data transmission service requests are required to be completed in the minimum possible time, more are required to be completed during a certain period of time; (2) meanwhile with little attention to: how to select node to optimize service cost, and with little consideration to network bandwidth constraint of requestor and the sensitivity of service requester to transmission time. If the grid system tries to ensure that each service request get minimum transfer time, it will certainly led to the increase of the overall service costs, so in peak time the acceptance rate and QoS will decrease dramatically.

An important problem now should be solved is: how to select replica node reasonably under the constraints of transmission time, to obtain a comprehensive objective of decision-making, including transmission speed, transmission distance, network status, requester bandwidth etc. So we can get a set of optimal service nodes, further we can get a multiple optimal objectives of decreasing network traffic, improving acceptance rate and QoS in peak, and guaranteeing a reasonable service costs. So a comprehensive decision-making model based on the status of grid system and the constraints of service requests is urgently needed.

II. BASIC CONCEPTS

A. Fundamental Analysis

For a parallel transmission service, first we should consider how to choose replica nodes. During this process we should comprehensively consider the distance between requester and replica node, the status of network links, the effective bandwidth that the replica node can provide and its status, and ensure the aggregated bandwidth that participating nodes can provide and the

doi:10.4304/jsw.6.12.2472-2478

effective bandwidth of requester are rational. Then we must configure reasonable transmission speed for each parallel transmission channel, thus, in the premise of satisfying the transmission time, we can optimize the overall service cost.

Network cost: During the process of parallel transmission, the applicant downloads data from different replicas by a certain speed, as shown in Figure 1. Data move in the channels has brought a load to the network, and result in a cost. If a link that a channel must pass is busy, then the cost will be great when the data crosses it, accordingly, the more data crosses this channel, the more cost will be.

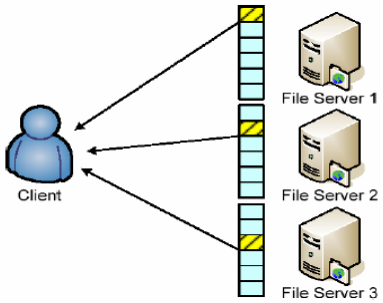


Figure 1. Striped transmission mode of GridFTP

Node cost: When replica nodes provide services to applicants, the service cost includes: connection cost and bandwidth cost. They both increase with time, and meanwhile the connection cost will increase with the amount of provided bandwidth also.

B. Symbol Definition

By the above analysis, the basic symbols used in model deduction are defined as follows:

TABLE Ⅰ. Key symbols used

M:	Size of data file;
k:	Sum of replicas;
N_i:	A replica node, (1≤i≤k)
B_i:	Maximum effective bandwidth from node N_i to service applicant
V_i:	Actual download speed from node N_i to service applicant;
V_a:	Actual bandwidth that the requester can achieve.
C_i:	Cost that transfer per unit data at per unit time from node N_i to requester;
W_i:	Cost that node N_i provides per unit bandwidth at per unit time;
A_i:	Cost for connection to node N_i at per unit time;
Q:	Effective maximum bandwidth of Applicant;
t_m:	Maximum transmission time constraint of applicant;
t_u:	Minimum transmission time that the applicant can achieve under current status of grid;
t:	Actual data transmission time;
z:	A positive which is greater than 0 and less than 1, is used to configure the lower limit of transmission speed from node N_i;
ω:	Weight factor for comprehensive optimization model, can be used to balance transfer time and service cost;
U:	Cost function for a data transmission service;
x_i:	0-1 decision variable, indicates that node N_i participates service or not;
y_i:	Decision variable, indicates that the actual transmission speed from N_i;
Cost:	Service cost;

C. Optimization objectives and basic constraints

Various algorithms based on GridFTP and multiple replicas, by assigning different data block from different replicas concurrently, make the requester download more data from the faster node. Its essence is to ensure each download process is uninterrupted, so as to obtain a maximum speed and minimum time.

Basic constraints:

(1) In a data service there must be some replica nodes involved in service and the others not. Let decision variables x_i represents that whether N_i is involved in service or not, here value '1' represents participance and value 0 represents not. See formula (1).

$$x_i = 1,0 \quad (1 \le i \le k) \tag{1}$$

(2) The effective transmission speed from N_i during a data service is defined as decision variables y_i. Here y_i must be less than or equal to the maximum effective link bandwidth from N_i, and it cannot be infinitely small, see equation (2) below.

$$zB_i \le y_i \le B_i \quad (1 \le i \le k, 0 < z < 1) \tag{2}$$

(3) The network bandwidth of requester has a certain threshold limit (Q), the actual transmission speed cannot exceed the threshold, see Equation (3). The threshold can be set according to the normal maximum bandwidth that the requester can achieve.

$$\sum_{i=0}^{k} y_i \le Q \tag{3}$$

(4) The minimum time for a parallel transmission service can be expressed as:

$$t_u = \begin{cases} M/Q & \sum B_i \ge Q \ \ (a) \\ M/\sum B_i & \sum B_i < Q \ \ (b) \end{cases} \tag{4}$$

(5) The maximum transmission time constraint of applicant (t_m), the minimum transmission time that the applicant can achieve under current status of grid (t_u), and the actual data transmission time (t) must meet the constraint of formula (5)

$$t_u \le t \le t_m \tag{5}$$

At the premise of satisfying the transmission time constraint, we should decrease transmission time and the whole service cost as much as possible. So that, the grid system can provide better service for more requests in peak time, and the acceptance rate and overall QoS can also be increased to a certain extent.

From the optimization objectives and the type of decision variables (x_i, y_i) we can know that the optimization model are '0-1' integer programming and linear programming problems.

III. RESOURCE SELECTION MODEL

In this section, we first present two basic optimization models, as they both have some limitations, and then based on them an extended comprehensive optimization model is proposed.

A. The minimum transmission time model (A)

Currently, most researches about parallel transmission based on *GridFTP* tend to minimize transmission time, and pay little attention to network bandwidth of requester. In this section we give a node selection model. The model takes minimized transmission time as objective, and the bandwidth of requester as constraint, see formula (6).

The minimum transfer time can be calculated by two situations, such as formula (4). When 4-(a) is met, minimum transfer time t_u=M/Q; and when 4-(a) is met, t_u=M/$\sum$B$_i$.

$$\min Z = \sum_{i=1}^{k} x_i B_i \quad s.t. \begin{cases} x_i = 0,1 \ \ (i = 1,2,...,k) \\ \sum_{i=1}^{k} x_i B_i \geq Q \end{cases} \tag{6}$$

The model can output the minimum transmission time, the set of participating nodes. But the model only seeks to minimize transmission time and ignore the impact of network load. The participating nodes may be far apart, the transmission path may be very busy, which will bring greater pressure on the network. At the same time the status of replica nodes are not considered too. Both factors will lead to excessively high cost.

As the optimized aggregated bandwidth is greater than threshold of requester, how to configure the actual transmission speed for every channel, in order to optimize the service cost, is not considered in this model too.

B. The Minimum Cost Model (B)

Reducing the cost of each service request can enhance the acceptance rate and Qos of grid system in peak time. If decision variable y_i is equal to 0, it indicates that node N_i does not participate in current service. If the download speed from node N_i is too small, it will decrease the utilization of replica node. Because the connection itself will cause some CPU overhead, it is necessary to set the lower limit for download speed, see equation (7)-d.

The minimum cost model is represented as formula (7). Optimization goal minZ consists of two parts: the transmission time $M/\sum_{i=1}^{k} y_i$ is the left part and the transmission cost at per unit time is the right part. By the meaning of y_i, C_i, W_i, A_i, the cost formula at per unit time can be represented as $\sum_{i=0}^{k} y_i (C_i + W_i) + A_i$. So the optimization equation (7)-a indicates the minimum service cost for a request.

$$\min Z = \left(M/\sum_{i=1}^{k} y_i\right)\left(\sum_{i=0}^{k} y_i (C_i + W_i) + A_i\right) \quad (a)$$

$$s.t. \begin{cases} M/\sum_{i=1}^{k} y_i \leq t_m & (b) \\ \sum_{i=0}^{k} y_i \leq Q & (c) \\ zB_i \leq y_i \leq B_i, \ i = 1,2,...,k & (d) \\ A_i = \begin{cases} A_i & y_i > 0 \\ 0 & y_i = 0 \end{cases} \ (i = 1,2,...,k) & (e) \end{cases} \tag{7}$$

Constraint equation (7)-b represents that the actual transmission time must meet the constraint of maximum transmission time (t_m) given by requester; constraint equation (7)-c represents that the aggregated bandwidth must be less than or equal to the current maximum bandwidth of the requester; constraint equation (7)-e represents that if the connection with N_i is established, the connection cost takes A_i, else takes 0.

This model outputs the minimum service cost and the actual transmission bandwidth from every replica node (also outputs the set of nodes involved in service). But with the constraints of time, the model will make the actual time (t) tend to t_m (maximum time), regardless of the sensitivity of the transmission time for applicants, this is very useful when the network is busy, but when the network is idle, we should try to make the actual time tends to t_u (minimum time).

C. Comprehensive optimization model (C)

Model A seeks to minimize transmission time, without considering the service cost and the applicant's transmission time constraints. Model B takes minimum service cost as optimization objective, it will lead to that the transmission time is always tends to longer time within t_m constraint. Therefore, a balance strategy should be established between transfer time and service cost. The applicant has a certain degree of sensitivity on transmission time, and there is also a certain relationship between the service cost and the state of the grid system. If the key level of the task running in requester side is high, then it indicates that the transmission time has higher priority, in contrary, if the grid system is busy, then it should have higher priority.

We can take transmission time and service cost as optimization objective at the same time, and set a weighting factor to balance them. Complete optimization models is as shown below. L represents the difference between the actual transmission (t) and the minimum transfer time (t_u); U represents the whole cost of a service; ω is a weighting factor which can be used to balance transfer time and service costs; other specific constraints can be found in equations (1) - (5).

$$\min Z = \omega L + U \quad (a)$$

$$s.t. \begin{cases} i = 1,2,...,k & (b) \\ S = \sum_{i=1}^{k} B_i & (c) \\ t_u = \begin{cases} M/Q & S \geq Q \\ M/S & S < Q \end{cases} & (d) \\ t = M/\sum_{i=1}^{k} y_i < t_m & (e) \\ L = t - t_u & (f) \\ zB_i \leq y_i \leq B_i & (g) \\ A_i = \begin{cases} A_i & y_i > 0 \\ 0 & y_i = 0 \end{cases} & (h) \\ U = \sum_{i=0}^{k} y_i (C_i + W_i) + A_i & (i) \end{cases} \tag{8}$$

The meaning of the linear programming model is: in the premise of satisfying the constraints, take the minimum value of Z, solve the values of decision variables y_i, and then we can further calculate the actual transmission bandwidth, transmission time t, and whole service costs.

IV. SIMULATION EXPERIMENT

There are many tools that can be used to solve '0-1' integer programming problem, such as *Matlab*, *Lingo* and *Lindo*. As *Lingo10* is capable of flexible input/output and programming, and is more flexible than Matlab in solving complex integer programming problem, so we use Lingo10 to solve the models. In experiment some data are generated randomly, we observe the impact of various parameters on decision.

Here we give a network including 8 nodes, where N_1~N_7 are deployed replicas, N_0 is a requester. The corresponding parameters are given in table 2.

TABLE Ⅱ VALUES OF B_i, C_i, W_i, A_i

Nodes / Paras	N_1	N2	N3	N_4	N_5	N_6	N_7
B_i	105	95	84	52	71	49	82
C_i+W_i	6	7	5	6	4	8	5
A_i	26	18	35	20	39	42	47

A. Purpose

Model A seeks to the minimum transmission time, and this will lead to higher cost to grid system. While in model B, the range of transmission time is given, so it must have impact on service cost. So we should check the effectiveness of model A and B, and compare the transmission time. In model C, a weighting factor is introduced to balance transmission time and service cost, so we should check its effectiveness also.

B. Experiment 1: test for minimum transmission time model

This experiment is designed to check the relationship between the aggregated bandwidth provied by the set of nodes output by model A and the maximum bandwidth of requester, and also to check effectiveness of model A. The values of B_i are given in table 3. By changing the values of Q, we conducted several experiments, and the experimental data is shown in Table 3.

TABLEⅢ. Decision-making results

Times	1	2	3	4	5
Q	50	70	90	110	130
Z	68	79	92	115	136
N_i	N_1, N_2	N_2, N_4	N_5	N_1, N_2, N_4	N_3, N_4

From figure 2 we can see that the decision-making results meet the constraints of the model, i.e., Z≤Q, and the difference between them is little. Therefore, the actual transmission bandwidth of requester from the participated nodes can reach to Q, and the minimum transmission time can be calculated by formula (4).

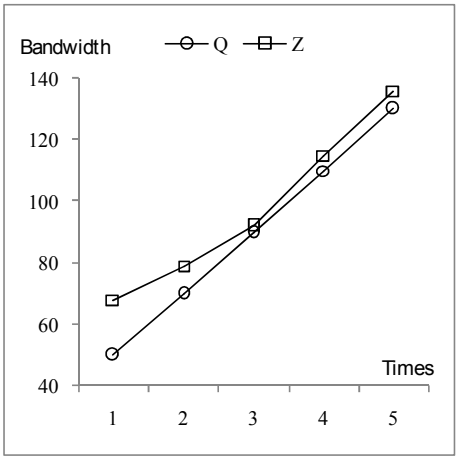


Figure 2. Comparison of Q and Z in model A

C. Experiment 2: Test for minimum cost model

(1) Performance analysis when M changes

This experiment is designed to check the relationship between t and t_m under specific condition when M is changed, and to check the relationship between V_a and Q. Let t_m=50, q=130, z=0.2, then observe a set of results for every value of M. The experimental data is shown in Table 4.

TABLEⅣ. Optimization results of model B when M changes

Times	1	2	3	4	5
M	2000	2500	3000	3500	4000
t_m	50	50	50	50	50
t	37.5	46.8	50	50	50
t_u	15.4	19.2	23.1	26.9	30.8
V_a	53.4	53.4	60	70	80
Q	130	130	130	130	130

It can be seen from Figure 3 that when M takes different values, the actual data transmission time t lies between t_m and t_u., and it is more closely with t_m, even overlap. The result is consistent with the constraints of formula (5). From figure 4 we can know that, the actual bandwidth achieved by requester is far less than its maximum bandwidth Q. So it leads to that the actual transmission time t is much larger than the minimum transfer time t_u.

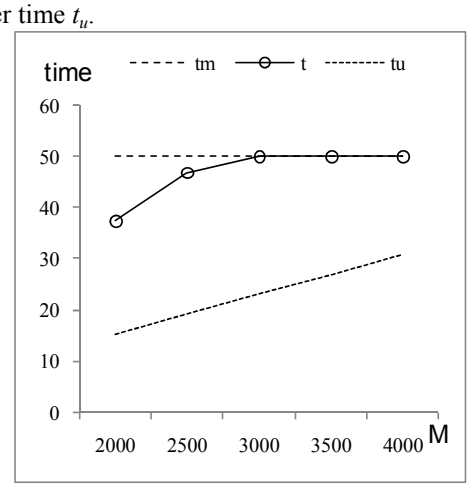


Figure 3. Impact of M on t in model B

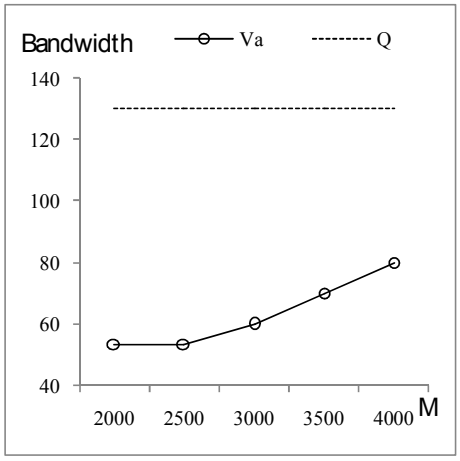


Figure 4. Impact of M on V_a in model B

Analysis: Since the minimum cost model at a given time constraints in order to achieve the minimum cost objective, will reduce the actual transmission bandwidth of the applicant. The state of grid is not considered in the optimization process. This can improve the quality of service and acceptance rate in peak time, but when the grid is idle, it will result in a large number of idle resources.

(2) Impact of t_m on the performance of model

This experiment is designed to check the impact of t_m on service cost. Maximum expected time constraint given by the applicant is an important basis for decision making, so for a service request we should check how t_m impacts on service cost. Let M=3500, q=120, z=0.2, then incrementally change the values of t_m, and every time record the result of service cost output by the model. Observe 5 sets of data, see table 5.

TABLE Ⅴ. Optimization results when t_m changes.

Times	1	2	3	4	5
t_m	30	35	40	45	50
cost	9931.25	7383.83	6620.65	5933.66	5816.90
t	30	32.5	35.8	43	49.3

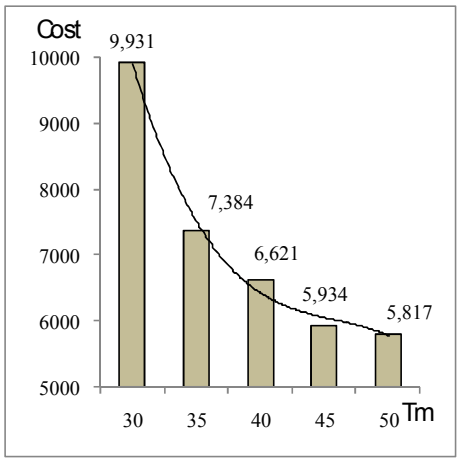


Figure 5. Impact of t_m on cost in model B

From figure 5 it can be seen that, as t_m increases, the whole service cost decreases, the difference is 4114 when t_m=30 and 50, and the cost is decreased by 41 percent. But by figure6, as t_m increases, t is always closely with t_m, the two curves is cross in the first time, and in the third the difference is a little more, but in the four and five times they gradually become close. This shows that under the constraint of t_m, the actual transmission time is still biased in favor of t_m.

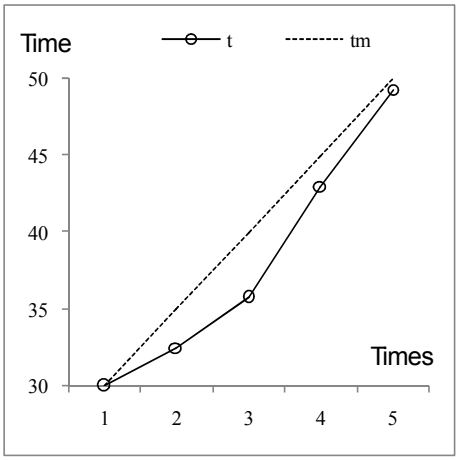


Figure 6. Impact of t_m on t in model B

D. Experiment 3: Test for comprehensive optimization model

Model A takes minimum transmission time as the optimization objective, and the service costs are not taken into account. In model B, the whole service cost is optimized under the constraint of expect maximum transmission time, so this leads to the actual transmission time tends to t_m. Therefore, in model C, a weighting factor is introduced to balance the time and cost. This experiment is designed to the check the trend of service cost and time when the weighting factor changes. Let M=3000, t_m=50, q=150, z=0.2, then change the value of ω and observe four sets of data, as shown in Table 4.

TABLEⅥ. the corresponding optimization results outputted by model C when ω changes

ω	40	60	80	100
t	45.2	28.3	24.0	20.0
t_u	20.0	20.0	20.0	20.0
t_m	50	50	50	50

It can be seen from figure 7, with the increases of weighting factor, the actual transmission time t which is close to the maximum transmission time (t_m) at the beginning, gradually moves to the minimum transmission time t_u. When ω=100, curves t and t_u intersects and achieves the minimum value. In this process, the curve gradually increases and the corresponding values increases to 27860 from the beginning value 19450, and the cost increases by 43.2%. From figure 8 we can see that as the increases of weighting factor, the service cost increases greatly, it just shows an opposite trend compared with curve t.

We can see that the weighting factor can balance transmission time and service cost effectively. If the applicant is sensitive to transmission time (or more

critical services), we can increase the weight, so under the constraint of time we can make the actual transmission time t tends to the minimum transfer time t_u. In contrast, if the critical level of service is general and the status of grid is busy, then we can reduce the weighting factor, thus prolong the transmission time and reduce service costs.

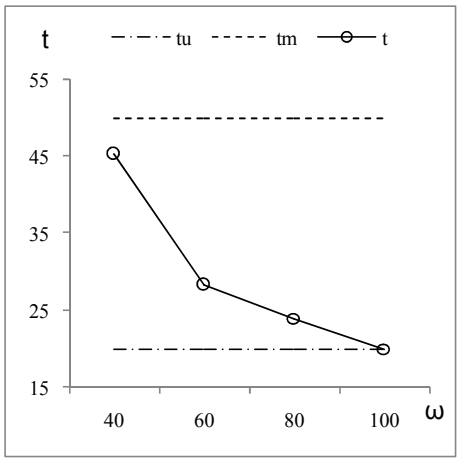


Figure 7. Impact of ω on t in model C

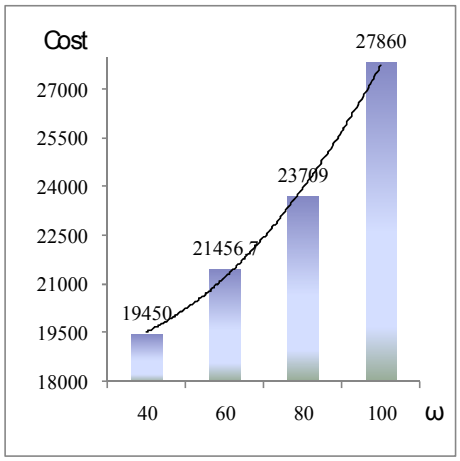


Figure 8. Impact of ω on cost in model C

E. Experimental Summary

For traditional parallel transfers that take minimized transmission time as optimization objective, a node selection model A is proposed. The model will cause a larger service cost, so the availability is better when the grid is free. In model B the range of transmission time is introduced, the model takes the minimized service cost as optimization objective, so compared with model A it has a greater advantage for the optimization of grid load. But the model always tend to longer time within time constraint regardless of grid status (idle or not), so it has a better usability when the status of grid is busy. In model C by introducing a weighting factor, the model not only focuses on service cost and transmission time, but also can make dynamically decision according to the critical level of transmission time and current status of grid. The weighting factor in model C palys a good role of balancing transmission time and service cost. Model C has better applicability and feasibility.

V. CONCLUSIONS

Guaranteeing the QoS of data service in grid is a challenge. Previous parallel transmission algorithms (based on multi-copy) less concerned about node selection and service cost optimization problem. The resource selection model (comprehensive optimization model C) proposed, under the condition of meeting the transmission time constraints, according to a variety of factors, can output the set of nodes involved in service. We achieved the optimization objective of balancing transmission time and service cost. Optimization of service cost and rational use of node resources can improve the acceptance rate and QoS of grid system in peak time. The comprehensive optimization model provides a useful reference for the guarantee of QoS for data grid and node selection.

REFERENCES

[1] Esther Pacitti. Patrick Valduriez. Marta Mattoso. Grid Data Management: Open Problems and New Issues. Journal of Grid Computing. 2007, 5:273–281

[2] Tim Ho, David Abramson. A Unified Data Grid Replication Framework. Proceedings of the Second IEEE International Conference on e-Science and Grid Computing: IEEE Computer Society, 2006:1-8

[3] Pangfeng Liu. Jan-Jan Wu, Optimal Replica Placement Strategy for Hierarchical Data Grid Systems. Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid:IEEE Computer Society, 2006:1-4

[4] William Allcock, John Bresnahan, et al. The Globus Striped GridFTP Framework and Server. Proceedings of the 2005 ACM/IEEE SC|05 Conference. Seattle, WA, USA:IEEE Computer Society, 2005, 1-11.

[5] Jun Feng, Lingling Cui, et al. Toward Seamless Grid Data Access: Design and Implementation of GridFTP on .NET. The 6th IEEE/ACM International Workshop. Vienna University of Technology, Austria:IEEE Computer Society, 2005,1-8.

[6] Chao-Tung Yang, I-Hsien Yang, Kuan-Ching Li. A Recursive-Adjustment Co-allocation Scheme in Data Grid Environments. ICA3PP 2005, LNCS 3719. 2005:1-10

[7] Chao-Tung Yang, I-Hsien Yang and Kuan-Ching Li. Improvements on dynamic adjustment mechanism in co-allocation data grid environments. J Supercomput,2007,40:269-280

[8] Chao-Tung Yang, Shih-YuWang, William Cheng-Chung Chu. Implementation of a dynamic adjustment strategy for parallel file transfer in co-allocation data grids. J Supercomput,2010,54:180-205

[9] Dafei Yin, Bin Chen, Yu Fang. A Fast Replica Selection Algorithm for Data Grid. 31st Annual International Computer Software and Applications Conference(COMPSAC 2007), IEEE, computer society, 2007:1-4

[10] Gaurav K, Umit C, Tahsin K, et al. A Dynamic Scheduling Approach for Coordinated Wide-Area Data Transfers using GridFTP. PARALLEL AND DISTRIBUTED SYSTEM, 2008,1-12

[11] Husni Hamad E. AL-Mistarihi and Chan Huah Yong. Response Time Optimization for Replica Selection Service in Data Grids. Journal of Computer Science,2008,4(6):487-493

Ming-Cheng Qu is a Ph.D. in school of Computer Science and Technology of Harbin institute of technology (HIT). He received his BS and MS degree from HIT. His research interests include grid computing etc

Xiang-Hu Wu is a professor in school of Computer Science and Technology of Harbin Institute of Technology (HIT). He is a advanced member of CCF. His research interests include grid computing and embedded computing.

Vector-Mapping Method for Motion Retargeting of the Virtual Articulated Figures and its Application

Xiao-juan Ban
University of Science and Technology Beijing, Beijing, China
Email: banxj@ustb.edu.cn

Dong-qin Han and Tian Jin*
University of Science and Technology Beijing, Beijing, China
Email: jintian@ustb.edu.cn

***Abstract*—Motion Retargeting is a movement edition technology that designed for motion capture data of the production of computer animation. The technology of Motion Retargeting now has two problems--large databases and the complexity of the algorithm increased rapidly with the joints increasing. According to this, we propose a new method, Vector-Mapping method of motion retargeting. The initial movement curve is transferred to vectorization and mapped to the target figure. The virtual figure is setting by IK, which moved along the trajectory curve in order to achieve motion retargeting. Finally, we implemented several experiments based on this method and described some ideal results in this paper.**

***Index Terms*—Motion Retargeting, vectorization, mapping, virtual arms**

I. INTRODUCTION

Motion Retargeting technology is a motion edition technology of new computer animation proposed firstly in the late 1990s by Autodesk's Gleicher[1].The basic idea is to capture the movement of data applications to a different joint model, while maintaining the original characteristics of the movement, re-use existing joint movement data. This idea can be applied with the same joint structure but different from the proportion of the virtual characters, and achieved good results. In recently years, many researchers began to study movement retargeting technology based on the "Motion Retargeting" mentality, from the same joint structure extend into the joints with a different topology objects. Through the movement of existing data [2] for Motion Retargeting, has made remarkable achievements. Monzani[3] and others proposed a combination of the use of intermediate skeleton and inverse kinematics model of the movement retargeting method, which proved a good structure with different joint movement between the virtual role of retargeting. According to the corresponding joint bone technology, Hsieh[4] etc implemented the motion retargeting in virtual characters with different joint structure, such as the motion retargeting between humans and dogs, and between humans and sharks. This makes the reusability of original motion data has been further strengthened.

Technology of motion retargeting is not new, but the existing methods require a large database to store the virtual role models and the collected data of initial motion [5], and the combination of inverse kinematics has been movement on the joint displacement and rotation information redirected to the target object in order to achieve the movement redirect[6]. The calculation process is complex, and with the joint increasing, exacerbated by the complexity of the algorithm redirects to analyze the cost of solving is also growing.

Aiming at the existing problem of motion retargeting that the shackle of a large database, as well as the complexity of redirect algorithm that relate with joint DOF, this paper presents a new way to redirect the movement, Vector-mapping method of Motion Retargeting.

On the basis of virtual role joint model with IK (Inverse Kinematics) settings, curve of the initial motion after vectorization and proportional adjustments, and then redirected to the target, the terminal effector of the target object drive the corresponding joint along the motion curves, so as to realize the motion retargeting. This method only needs to store the initial motion curve, and solve the problem of a large database in a certain extent and the algorithmic complexity doesn't increase when the joints increase.

II. DESIGN FOR VECTOR - MAPPING METHOD OF MOTION RETARGETING

Currently, retargeting technology mainly applies in two aspects [7]: 1) virtual character with joint connection

*Corresponding author, School of Computer & Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China.
Tel.: +86 13810508419
E-mail address: jintian@ustb.edu.cn

doi:10.4304/jsw.6.12.2479-2485

in character animation, 2) facial expression animation. In this paper, the vector-mapping algorithm of motion retargeting only applies to the first case, and the object must be virtual role with joints of IK settings, which is the premise of this method can be realized.

A. Modeling of Virtual Role

Virtual character modeling includes two parts, namely, geometric modeling and motion modeling. Geometric modeling bases on the physiological characteristics of natural biological structure of its skeleton modeling; the method in this paper only applies to role model with joint structure. From the kinematic point of view, joint information of moving objects can be simplified, under the premise of ensuring the fidelity of movement. For example, the virtual arm of our experiments in this paper, the joint structure can be simplified as: shoulder, elbow and wrist joints. In order to reflect the vector-mapping method for the movement curve of the operation better in this experiment, joint structure is simplified as shoulder and elbow joint. (Fig.1) (a).

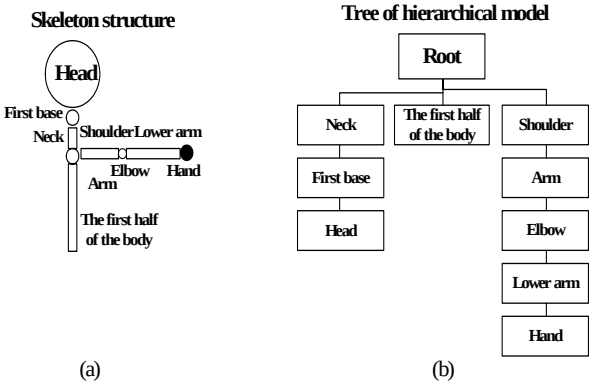


Figure 1. Modeling of virtual role.

B. Vector-Mapping Algorithm for Motion Retargeting

The overall idea of Vector-Mapping algorithm for motion retargeting is to set moving object based on IK, read the original motion curve, extract sharp points and vectorize, zoom and adjust the motion curves converted to the motion curves of a re-set target object, finally the motion curves are mapped to the terminal effectors of the target object. According to the trajectory of motion curves, the terminal effector completed a redirect movement. Using vector-mapping to achieve computer animation of the virtual role in the movement retargeting issues, we need the following three steps:

1) The initial motion curves

From the motion capture technology to capture the movement of data generated by computer animation, which is low efficiency, costly and not conducive to the development of computer animation. In this paper, we introduce a virtual scene movement of an object as a reference, tracking, extraction of its trajectory, that is, the initial motion curve. Specific method is to record a animation sequence, generally under 50 frame, which basically guarantee for a full period of motion. The acquisition of the initial motion curve requires the following two-step [9]: (1) Moving target detection: to take "Background subtraction method" to get the moving objects' terminal effector .(2) Moving Target Tracking: to predict the goal of the future with Kalman filter, with the goal of cost function-based matching, tracking and recording the terminal effector 's movement, which is the initial motion curve. Simultaneously, record the size of capture object and this movement joints chains. Generally, Capture object is the standard model, in order to adjust different backbone of the ratio of virtual objects in motion curve.

2) Vector-mapping algorithm for motion retargeting

Step 1: read the initial motion curves;

Step 2: according to the vectorization algorithm, calculate the initial motion curves. For random, uncertain motion curves, the vectorization algorithm can be very good to preserve the data information and the characteristic of original motion curves.

Step 3: determine the beginning and the ending of the motion curves;

Step 4: calculate the ratio of backbone between the direct target object and the capture object, the capture object is standard model normally. According to the value of the ratio of backbone, zoom the motion curves corresponding, convert to the motion curves of the direct target object. When the size and the proportion of the target object do not conform to the standard model, its movement may lose the original characteristic, or go against nature, so zoom and adjust the motion curves is very necessary.

Step 5: in the virtual environment, virtual character's position is not affirmatory, in order to ensure that the data information of motion will be mapped to the target object, we must determine the position of terminal effector of the target object, and unified the coordinate system of the starting point of motion curves and the terminal effector of the target object.

Step 6: determine the surface of movement of direct target object. In 2D situation, the surface is plane still, in 3D situation, determine the movement surface in line with the movement characteristics.

Step 7: according to the principle of line-face mapping, we can mapped the data information of target object to the terminal effector.

3) Complete motion retargeting

Direct target object is the joint model with IK settings, given the target location of terminal effector, computer will calculate the data information of joint automatically. Motion curves is a series of target points of terminal effector, along the trajectory of motion curves, the direct target object completed the whole movement.

III. VECTOR - MAPPING ALGORITHM FOR MOTION RETARGETING

A. Cusps Extraction of the Initial Motion Curves

The initial motion curve with high randomness and uncertainty, in order to avoid losing their original characteristics and efficient to save the key feature points, we must first extract the tip points, namely curvature extreme point. In this paper, we use the method of chord

length ratio[10] to calculate the curvature of the curve and look for the local maxima and mark the point for cusps(Fig.2). Specific steps as follows:

Step 1: extract the point $P_i(x_i, y_i)$, $P_{i-k}(x_{i-k}, y_{i-k})$, $P_{i+k}(x_{i+k}, y_{i+k})$ in motion curves;

Step 2: according to two point's form of linear equations, we can obtain the straight line equation which connects P_{i-k} and P_{i+k}

$$Ax + By + C = 0 \tag{1}$$

And $A = y_{i-k} - y_{i+k}$; $B = x_{i+k} - x_{i-k}$; $C = x_{i-k}y_{i+k} - x_{i+k}y_{i-k}$

Step 3: chord length L_{ik} between the point $P_{i-k}(x_{i-k}, y_{i-k})$ and $P_{i+k}(x_{i+k}, y_{i+k})$

$$L_{ik} = \sqrt{(y_{i-k} - y_{i+k})^2 + (x_{i-k} - x_{i+k})^2} = \sqrt{A^2 + B^2} \tag{2}$$

Step 4: the distance of the point $P_i(x_i, y_i)$ to the chord L_{ik} mark as

$$d_{ik} = \frac{\left|Ax_i + By_i + C\right|}{L_{ik}} \tag{3}$$

Step 5: base on the method of chord length ratio we can obtain the curvature of the curve in the point P_i :

$$c_{ik} = \frac{d_{ik}}{L_{ik}} \tag{4}$$

Step 6: if c_{ik} is greater than or equal to threshold, mark $P_i(x_i, y_i)$ as cusp;

Step 7: turn to step 1 until another endpoint of the curve

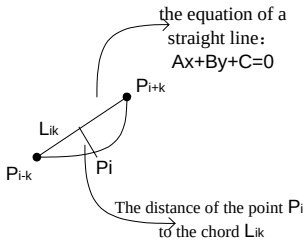


Figure 2. Schematic of chord length ratio.

B. The Fitting- Vectorization Calculation of Tangential Line and Straight Line

With high randomness and uncertainty, the initial motion curves must save its data information to meet the request of vector calculation. For this feature we choose the vectorization algorithm of curve-line fitting based on tangential line [11]. The main purpose is to use the fact that the tangent of the curve can reflect the changes of its curvature, and the distance of the point to the tangent which can be calculated in standard vectorization. The curve segment at a certain distance within the threshold can be fit with this straight line, and curve segment greater than the threshold can be fit with another tangential line. The idea of using this method of the fitting- vectorization calculation of straight line as follows:

Set the minimum point of the initial motion curves in the direction of *x* as starting point e_{start} and e_{end} as the other side of the end for the movement.

Draw the tangential line L_{e1} of curve from e_{start} , and then calculate the distance d of the point in the curve to the tangential line L_{e1} ,if $d = D$ (D as the threshold),mark this point as p_1 ;And then draw the tangential line L_1 through p_1 ,if $d = D$ mark this point as p_2 ; By analogy, draw the tangential line L_n through p_n ,when $d = D$, mark this point as p_{n+1} , until the distance $d \le D$ that finishing point e_{end} to the last tangent L_m ;Use straight line connect the points $\overline{e_{start}p_1}, \overline{p_1p_2}$, ……$\overline{p_ne_{end}}$,namely complete the fitting- vectorization of curve(Fig.3).

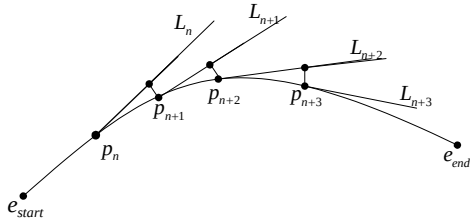


Figure 3. Tangential line and straight line fitting schematic.

C. Mapping Algorithm Based on Scaling Transformation

After the initial motion curves optimization-calculation, we get a series of athletic data point information, and connect the data points that are motion curve after optimization. The movement information will be mapped to the target object to finish motion retargeting process. However, before mapping the data information, we need to calculate the proportion of backbone between the reset target object and the standard model. According to the proportion of the backbone, we must zoom the motion curve after optimization corresponding, convert it to the motion curve of target object and then to be mapped. So we can avoid distortion of motion retargeting between models with different proportions of backbone.

In 2D circumstance, the movements of the reset target object are in 2D plane. With the increasing degree of freedom in the 3D case, freedom of movement of virtual roles objects is also increase, we need determine the curved surface of movement of the target object and then the motion curve mapping to the curved surface of movement, so as to finish motion retargeting. According to different situations between 2D and 3D, we adopt a mapping mode which based on scaling transformation [12]: The motion curve after vectorization and the target plane of motion do the same mapped meshing, adopt mapping mode which based on scaling transformation. The change of proportion is divided into three cases in Fig.5. Algorithm flow chart shown in Fig.4.The plane of motion curve after vectorization and curve and the target

plane of motion do the same mapped meshing. So, all of them have the same quantity of grids.

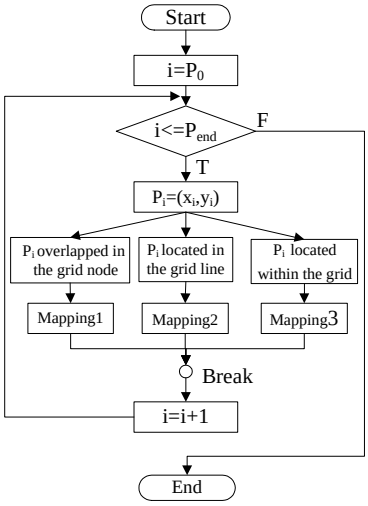


Figure 4. Flow chart of algorithm based on scaling and mapping.

Mapping 1: $P_i(x_i, y_i)$ overlapped in the grid node (Fig.5 (a))

Determine the row and column of grid line where original point located in;

According to the same row and column, we can get the mapping point in the target plane of motion;

Mapping 2: $P_i(x_i, y_i)$ located in the grid line (Fig.5 (b))

Determine the row and column of grid line where the original point located in;

Find the grid line corresponding to the target plane of motion, and calculate a, b and the arc length b1;

According to the relationship of scale a/b = a1/b1, determine the mapping point in the target plane of motion corresponding.

Mapping 3: $P_i(x_i, y_i)$ located within the grid (Fig.5 (c))

Set up parameter curve according to the original point in the surface (as shown);

Determine the grid where the original point and its mapping point located in;

Calculate AB, AC; DE, DF; BO, BE; A1C1 and D1F1;

According to the proportion AB/AC = A1B1 / A1C1 and DE/DF＝D1E1/D1F1 determine the points B1and E1 inflattening map;

Extract the points B1 and E1 in flattening map, and generally, these points have different v-direction parameter values, calculate their average value v0, and through v0 draw a new u-direction curve, which has the same points B1'and E1'with the grid line;

Calculate the length of the arc B1'E1', with the u-direction curve crossingB1'and E1', we can determine the corresponding point O1 of O according to the scaling relation BO/BE＝B1'O1/B1'E1';

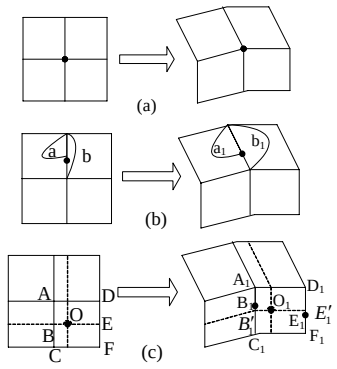


Figure 5. Three cases of scaling transformation.

IV. VECTOR - MAPPING ALGORITHM FOR MOTION RETARGETING EXPERIMENTAL RESULTS

A. MatLab Experimental Simulation

Following the idea of vector - mapping algorithm of motion retargeting, computer can calculate the information of its intermediate joint automatically So in MatLab simulation, we can just check whether the motion curve or the feasibility of our method in computer animation can be verified.

Because we need to extract the information of motion curve, and design it again, for random trajectory of motion curve we adopt the method of extracting pixels and its movement information.

As shown in Fig.6. Then do the fitting- vectorization calculation for initial motion curve by using least-square method. Besides computing easily, using this method can maintain the characteristic of original curve.

In a new window displays the motion curve of vectorization, and a fixed point of reset target object, as shown in Fig.7, in the 2D circumstances, the situation that fitting vectorization and mapping of the initial motion curve. In Fig.8, in 3D circumstances, the situation that fitting vectorization and mapping of the initial motion curve.

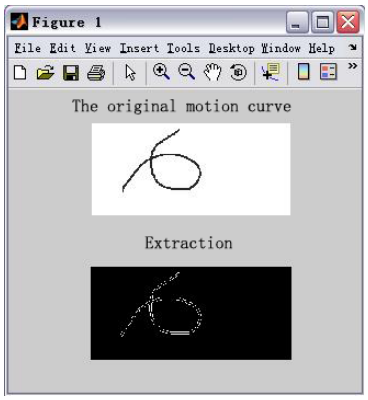
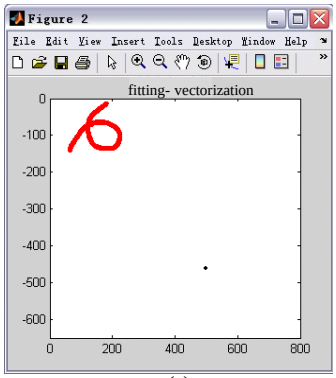
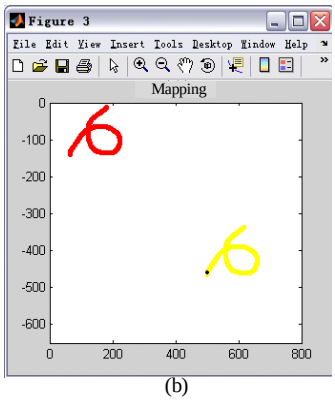


Figure 6. Read and extract the initial motion curve.

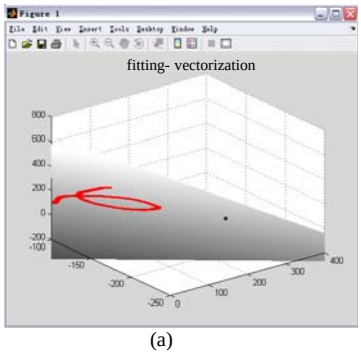


(a)

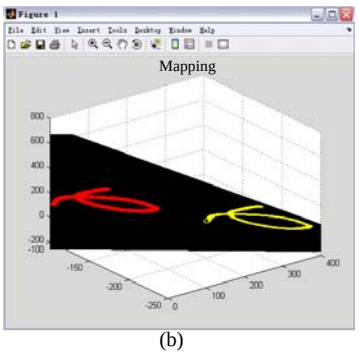


(b)

Figure 7. 2D fitting vectorization and mapping.



(a)



(b)

Figure 8. 3D vectorization and mapping.

After MatLab experiment, for any motion curve, the method of vector-mapping is good and the result is precise. Because the method of vector - mapping is for virtual object with IK settings, and the object's terminal effector can drive corresponding joint moving together. If the motion curve can vector-mapping correctly, thenthe motion retargeting can finish. After the simulation above, the method of vector-mapping proposed in this paper is feasible. Next, we study the concrete object- virtual arm to verify our method in virtual environment.

B. Using the Virtual Arm to Verify Vector-Mapping Algorithm of Motion Retargeting

This experimental simulation adopts Visual c++ 6.0 &OpenGL. Use the method of vector-mapping, by taking two virtual arms with identical skeleton structure but different lengths as research objects.

We use the random motion curve to verify motion retargeting between the two virtual objects and get a good result. The experimental simulation adopts MFC frame structure. In the framework generated by MFC, the movement and motion retargeting of virtual objects in work area can be achieved.

The virtual object in experiment is virtual arm, which uses the HI settings. Define a structure T_Bone , set the hierarchical relationship of father-son between bones, and define the proportion of backbones, the length of the bone, and rotate, translate variable, and then use the algorithm of IK Solver to set up the virtual arm.

Motion retargeting between the two virtual arms has some characters as : In virtual scene, the position of mouse directly relate to the position of Model-1 arm terminal. If the target points in the reachable area, terminal effector will move to the target point based on inverse kinematic, Track the position of mouse to complete a movement and record the movement of the curve. Then this movement can be mapped to Model-2 virtual arm, and the virtual arm moves along the motion curve. Motion retargeting between the two virtual arms is completed. As shown in Fig.9:

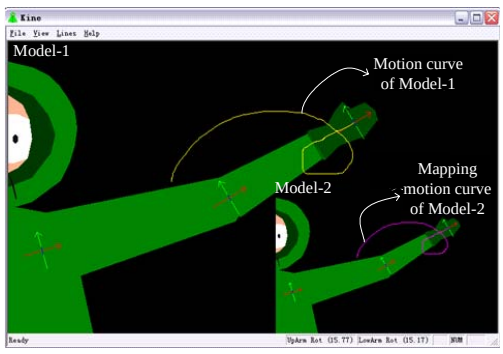


Figure 9. Motion retargeting between the two virtual arms.

Another mapping method is motion curves mapping, namely reading motion curves, and we use the vector - mapping algorithm of motion retargeting to extract the cusps and data information of the initial motion curve, and then mapping to Model-1 and Model-2. Model-1 is standard model direct mapping, and Model-2 as a small scale model must map after scaling. The movement of virtual arm has the problem of reach ability, we must limit the area of mapping motion curve, and only in the area within the scope the point can be effective. Model-1 and Model-2 move along their own motion curve. As shown in Fig.10.

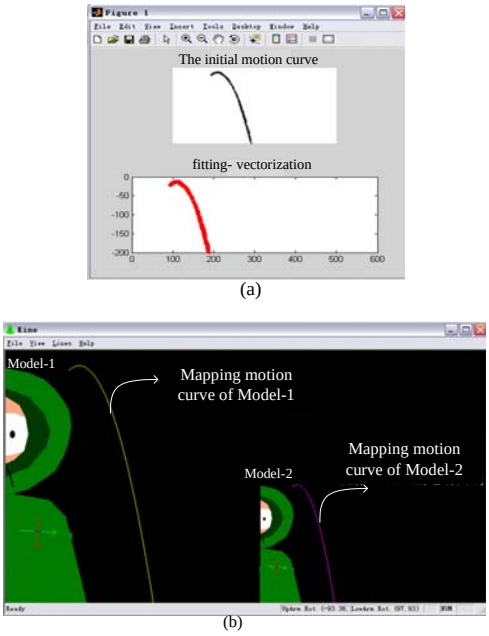


Figure 10. From the motion curve to Motion retargeting between the two virtual arms.

V. CONCLUSIONS

The IK Solver for 3D Max set the joints connecting to the model, and the vector-mapping method of motion retargeting is proposed in this paper, take the two virtual arms with identical skeleton structure but different backbone proportion as research objects, our method can be verified and we can receive a good result.

The idea of motion retargeting is reuse the existing data and information as much as possible. Also, we can use our method in a much wider range of areas. Based on the joint model with IK settings, we can use our method to achieve motion retargeting in objects with different skeleton structure and different backbone proportion. Our method is applicable between a role object with a skeleton structure and an object without having a skeleton structure. If all kinds of objects have the relationship of hierarchy, all objects can be considered as bone. Using our method can further strengthen the reusability of existing data, save the time and the cost of computer animation production.

ACKNOWLEDGMENT

This work wasSupported by National 863 Program(No. 2009AA062801)，National Nature Science Foundation of P. R. China (No.60973063)，Beijing Natural Science Foundation of P.R. China (No. 4092028)，the Fundamental Research Funds for the Central Universities(FRF-TP-09-016B) and supported by the new century personnel plan for the Ministry of Education(NCET-10-0221).

REFERENCES

[1] Michael.Gleicher, “Retargeting Motion to New Characters,” International Conference on Computer Graphics and Interactive Techniques,1998.

[2] Weidong Geng, Gino Yu, Reuse of Motion Capture Data in Animation: A Review. Springer-Verlag Berlin Heidelberg,2003.

[3] Jean-Sebastien Monzani,Paolo Baerlocher, Ronan Boulic and Daniel Thalmann, “Using an Intermediate Skeleton and Inverse Kinematics for Motion Retargeting,”Computer Graphics Forum,2000.

[4] Ming-Kai Hsieh, Bing-Yu Chen, Ming Quhyoung, “Motion Retargeting and Transition in Different Articulated Figures,” Proceedings of the Ninth International Conference on Computer Aided Design and Computer Graphics,2005.

[5] Akanksha, Huang Z., Prabhakaran B., and Ruiz, Jr.C. R., “Visualizing animation databases,”International Journal of Software Engineering and Knowledge Engineering,2003.

[6] Behzad Dariush, Michael Gienger, Arjun Arumbakkam et al. “Online and markerless motion retargeting with kinematic constraints,”Intelligent Robots and Systems,2008. IROS 2008. IEEE/RSJ International Conference.

[7] Akanksha, Zhiyong Huang, B. Prabhakaran, Conrado R. Ruiz, Jr, “Animation toolkit based on a database approach for reusing motions and models,”Multimedia Tools and Applications,2007.

[8] 3ds max 2009 Role Binding Tutorial, http://www.verycd.com/topics421782.

[9] YIN Yan, GENG Zhao-feng, “Background Model-Based Moving Object Detection and Tracking,”Microcomputer Information,2008.

[10] C.H.Teh and R.T.Chin, “On the detection of dominant Points on digital curves,”IEEE Trans, Pattern Anal.Mach.Intel,1989.

[11] XIONG Zheng, ZHOU Zhao-hui, LU Ye, “A Tangent-based Algorithm to Line Fitting for Curves in Vectorization,”Computer Simulation,2008.

[12] MAO Xin, WANG Kun, HUANG Ying, CHEN Shu-ping, “Shoe Last Model Reconstruction and Surface Development Based on Reverse Engineering,”Journal of Engineering Graphics,2008.

Xiao-juan Ban received her Ph.D. degrees in Control Engineering and Control Theory from University of Science and Technology Beijing in 2003. Her research interests include artificial intelligence, artificial life and its application in computer animation..

She is currently a Professor of the Dept. of Computer Science and Technology at University of Science and Technology Beijing.

Prof. Ban is director of Beijing Institute of Artificial Intelligence and commissioner Intelligent Automation Chinese Association of Automation Committee. Her email is banxj@ustb.edu.cn.

Dong-qin Han received the master degree in Computer Science and Technology from University of Science and Technology Beijing. She majors in Artificial Intelligence.

Tian Jin received the master degree in Communication and Information System from University of Science and Technology Beijing. She majors in Artificial Intelligence.She is an associate engineer in Teaching Experiment Center.

Maximum Portfolio: A Query Condition Optimization Method

Zhi Yang
State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China
Email:yangzhi9721@yahoo.com.cn

Budan Wu, Junliang Chen, Pingli Gu
State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China
Email: {wubudan, chjl }@bupt.edu.cn, gplqy98@163.com

***Abstract*—Web Service is becoming the next generation of web-based application. With enhancement of quality of services and increasing quantity of services, service discovery becomes important. In the process of discovery, although the user is not familiar with service interfaces, he wants to obtain the best services with some limited simple keywords. Because the keywords are often incomplete or wrong or fuzzy to match with the services in service library, it is difficult to get the satisfying services. This paper provides an approach named as Maximum portfolio to deal with the problem. The aim of maximum portfolio is that the suitable composition is found out from ambiguous input keyword set. The improved service matching algorithm is proposed and applied to the Service Generating Platform. The result of experiment shows the algorithm can improve efficiency of service discovery.**

***Index Terms*—Service Discovery Process and Methodology, Maximum Portfolio, Service Matching Algorithm, Ontology**

I. INTRODUCTION

Web Service is one of implementation modes of distributed systems and becoming the next generation of web-based application. Available Web services are described and advertised, and then they can be discovered and composed by other applications to create new value-added systems [1]. Web services facilitate dynamic formation of Web-service-based software applications according to user's demand [2]. So they are becoming popular and widely accepted due to their accessibility and compatibility [3]. In SOA (Service-Oriented Architecture) , Web service discovery is extremely important. In the process of discovery, the query keyword array is the start of service matching algorithm. Although the user is not familiar with service interfaces, he wants to obtain the best services with some limited simple keywords. Because the keywords are often incomplete or wrong or fuzzy to match the services in service library, it is difficult to get the satisfying services. So how to find out suitable services with the keywords is very important and urgent.

Recently, in the service matching algorithms the keyword array is treated as accurate [6, 7, 8 and 9]. In the algorithm in UDDI [6], it is hypothesized that the keywords are available and accurate. In the algorithms in literature [9], the keywords are divided into IOPE, but it does not treat the accuracy of keywords. Our earlier work [7] which provides a new service matching algorithm did not consider whether the keywords are accurate. The algorithm's premise is that keywords are correct.

Existing Web service matching approaches pay attention to design and realize algorithms while the query keywords input by the user are ignored. Because that the interface descriptions of Web services are often terse and cryptic [5] and that the user does not understand the detailed interfaces of Web service, it is very difficult for the above approaches to solve the above problem. How to deal with the problem is the concerned content of this paper. A solution scheme is proposed. We make use of the idea and approach of maximum portfolio to find out the user's request. Using this method, it is not difficult to solve the above issue. Then we propose an improved service matching algorithm. And in detail, the performance of the matching algorithm is analyzed.

This paper is organized as follows. Section 2 defines the maximum portfolio. An improved service matching algorithm is proposed and analyzed in section 3. Section 4 verifies this algorithm by an experiment. The last section draws to a conclusion.

II. DEFINITION OF MAXIMUM PORTFOLIO

User's input information is inaccurate. So in order to obtain the user potential requirement, the theory of maximum portfolio is provided. The definition of maximum portfolio and portfolio granularity are shown as follows.

Definition 1 Maximum Portfolio: There exists a set $S = \{I_1, I_2, ..., I_n\}(n = 1, 2, ...)$ and there is a

doi:10.4304/jsw.6.12.2486-2494

constraint condition C. $M = \{I_{j_1}, I_{j_2}, ..., I_{j_m}\}(1 \le j \le n, m \le n)$ is a subset of S， and M satisfies C condition. That is, the composition of elements of set S can satisfy C. $R = \{M_1, M_2, ..., M_n\}$ is the set of all the compositions, in which $M_i(i = 1, 2, ..., n)$ can satisfy C condition. Then $MAX\{|M_1|, |M_2|, ..., |M_n|\}$ indicates maximum portfolio.

Definition 2 Portfolio Granularity: In above definition, the element number of composition $M = \{I_{j_1}, I_{j_2}, ..., I_{j_m}\}(1 \le j \le n, m \le n)$ is described by PG. Its value is

$$PG = |M| = |\{I_{j_1}, I_{j_2}, ..., I_{j_m}\}| = m \quad (1)$$

Note that

a. Maximum portfolio describes the relations among elements of set S. In a set $S = \{I_1, I_2, ..., I_n\}(n = 1, 2, ...)$, according to some condition some elements are together and form a new set. The set including the most elements that satisfies the condition is maximum portfolio.

b. Portfolio granularity is an index to describe portfolio degree. By means of this measure index, it is easy to compare among more than one composition.

Generally, in the set which includes n elements, there are three types of portfolio granularity (PG) as follows:

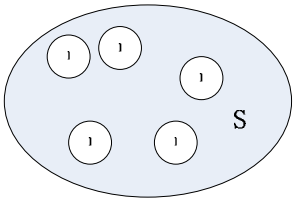


Figure1: PG=1

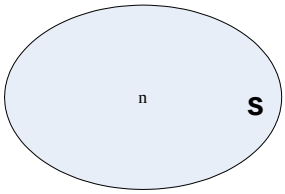
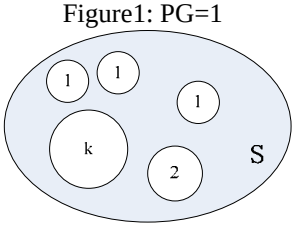


Figure2: PG=n



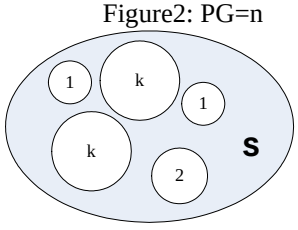


Figure3: PG=k

(k) indicates the set whose portfolio granularity is k($k = 1, 2, ...n$).

(S) indicates the whole set S.

Note: from the above figures, we can see that the biggest maximum portfolio may be all the element of the set (Figure 2) . Namely, set S is a composition and the composition is biggest. All the elements are in one area. The smallest maximum portfolio is one element (Figure1) . That is, there is no relation in any two elements of the set. The common maximum portfolio is shown as figure 3. There is one maximum portfolio in the left figure and more than one maximum portfolios in the right figure.

When maximum portfolio is applied to projects, one critical and challenging problem is how to obtain the maximum portfolio.

Two approaches are recommended as follows:

(1) The first step is that we examine one by one if each keyword of input parameter is effective. If with some keyword we can not get the available query result，then the keyword is deleted from the input parameter set.

The second step is that two keywords are chosen from the input parameter set every time. Then we justify whether the two keywords are effective. If they are available, we store them in the memory space M. Until all the two-keyword compositions are justified, we check the memory space M. If the M is not empty, the elements of M are candidate maximum portfolio. If the M is empty, we go on with the third step.

The third step is that three keywords are chosen every time. Then we justify with the three-keyword composition whether an available query result set is obtained from the ontology library. If the result set is not empty, the elements of memory space M are deleted and the three-keyword composition is put into M. Otherwise, we go on justify other three-keyword composition. Until all the compositions are judged, we check the memory space M. If the composition in the M is two keywords, the compositions in the M are maximum portfolios. Otherwise, we increase number of composition and repeat the third step, until the number of composition is equal to the number of the elements of input parameter set.

(2) The second approach is process of opposite compared with the first approach. It is that the number of elements of input parameter set is n and memory space M is created. The fist step we judge whether the whole input parameter set is available. If it is effective, the whole set is maximum portfolio. Otherwise, we go on with second step.

The second step is that n-1 elements are chosen every time. Then we judge if with the n-1 elements whether an available query result set can be obtained. If it is available, it is stored in the memory space M. Otherwise, another n-1composition is chosen and judged. Until all the compositions are judged, we check the memory space M. If M is not empty, the compositions in M are maximum portfolios. If M is empty, we carry on the next step.

The third step is that n-2 elements are chosen from the input parameter set every time. Then we judge if it is effective. If it is available, it is stored in the memory space M. Otherwise, another composition is chosen and judged. Until all the compositions are judged, we check the memory space M. If M is not empty, the compositions in the M are maximum portfolios. If M is empty, we decrease the number of composition and repeat the third step, until the number of composition is equal to 1. The figure of the process is as follow:

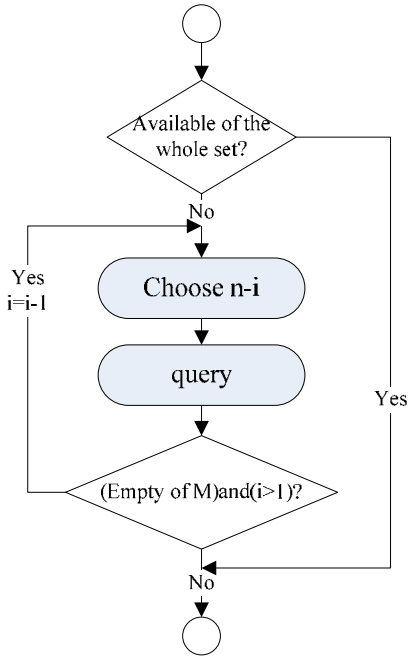


Figure4: The Process of Maximum Portfolio

From the two approaches, we can see that how to obtain the maximum portfolio of a set is a complicated and critical problem. Because that in the second approach, the first satisfying portfolio is maximum portfolio, in this paper the second approach is adopted.

By means of the thought of maximum portfolio we design a Filter to achieve maximum portfolio from the set of user's input keyword array. The process is showed as follow.

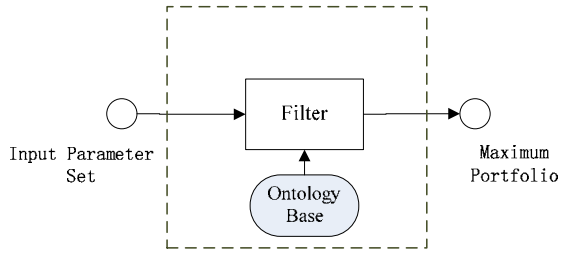


Figure5: Input Parameter Filter

Note that the input is user's input keyword set which is user's request. The output is maximum portfolio. Because it may be more than one, sometimes maximum portfolio is a set. The figure indicates using the ontology library the Filter extracts maximum portfolio set from input parameter set.

III. Improved Service Matching Algorithm

Based on our earlier work, an improved service matching algorithm is provided combined with the maximum portfolio. Firstly we pretreated user's keyword array in ontology library to obtain the maximum portfolio. Then by means of matching algorithm accurate result set is got.

The theoretical knowledge about service matching algorithm is introduced as follows.

A. Query Rewriting [10]

In the set of database relation, $T=\{T_1,T_2,...,T_n\}$ and its view set $V=\{V_1,V_2,...,V_n\}$, query Q are about set T of database relation. If there is a query Q_1 which searches at least a view in the view set V. Moreover, query result of Q_1 is consistent with query result of Q. So we claim that Q_1 is the query rewriting of Q.

B. Ontology

Ontology is a very important semantic technology. It is a description of the objective concepts and relationships [11].Ontology was originally a philosophical concept [12]. In 1998, Studer et al. further studied ontology on the base of study of predecessors and provided that ontology is a clear formal specification of shared conceptual model [13]. As a tool for knowledge representation, ontology structures the relations between knowledge points and provides a description or explanation of the domain knowledge to access the knowledge in a field.

Ontology can be described by OWL (Web Ontology Language) . OWL is the standard of ontology description language recommended by W3C. In order to solve semantic interoperability problem, explicitly it expresses the meaning of gloss and terms and their interrelationships [14]. OWL consists of three parts:

Individual: is the object which we are concerned about in some area;

Property: is a binary relation between individuals. That is, individuals create relations through properties.

Class: is a set of individuals.

Using the formal description method, OWL describes relationship between class and the members of class. At present, database is a very important storage approach of ontology and ontology is organized in the database according to some strategy and accessed by means of manipulation and management capabilities of existing database. As the relational database technology matures, most existing works of ontology data management take database management system of relation or object-relation as back storage. Currently popular relational database stores ontology in the database without losing the semantic.

By use of database storing ontology, there is an obvious advantage. That is, factually the operation of database table is the operation of ontology. Accessing ontology is equivalent to accessing database. So ontology library and database become one.

Generally, ontology can be described with some formats. For example, the following figure is about ontology hierarchical structure.

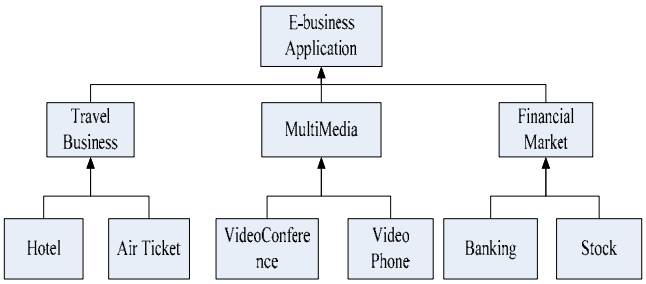


Figure6: domain ontology structure

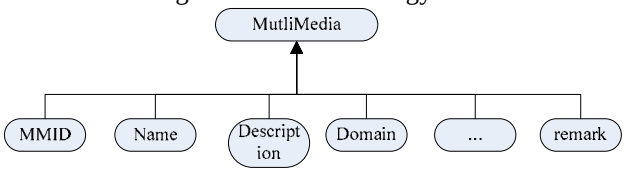


Figure7: the individuals of MutliMedia ontology

Note that the E-businessApplication can be classified into TravelBusiness, MultiMedia and Financial Market. MultiMedia can be classified into VideoConference and Video phone. MutliMedia consists of MMID, Name, Description, and so on. From the figure, we can intuitively see the links between individuals and hierarchical relation. By means of the theory of the relationship between part and whole of ontology, this paper provides an improved service matching algorithm.

C. Service Matching Algorithm

In this algorithm, firstly, the preprocessing is done to the user's input keyword array and we can obtain the maximum portfolio. Then query rewriting by use of ontology is done and in this case we can change query from keywords query to ontology query. The main process is divided two stages: grammar query stage and semantic query stage. After getting user's maximum portfolio requirements, we can analyze logical relationship and find out potential ontology information. Then we search ontology detail information in the database. In the end, we take ontology as query conditions to search data in the database.

First stage: (Grammar query stage)

Obtains the maximum portfolio and eliminates independent items. We adopt the second approach of generating maximum portfolio of set and obtain the maximum portfolio. Then the preprocessing of database set $T = \{T_1, T_2, ..., T_n\}$ and view set $V = \{V_1, V_2, ..., V_n\}$ can get rid of independent records with query and reserve the query-related records. The view set $V' = \{V_1', V_2', ..., V_n'\}$ is got for the second stage.

Second stage: (semantic query stage)

(1) Rewriting query. The user query conditions are taken as a part of the ontology. By using part of ontology in the ontology library the other parts of ontology can be obtained. That is, through the part we can get the whole. The user query is divided into $q(X) :- X_1, X_2, ..., X_n$. Among them, $q(X)$ is logical head which indicates the whole query, $X_1, X_2, ..., X_n$ is the body of conditions (usually, the conditions are not complete) , which is used to search the other parts of ontology in the ontology library. We get the individuals of ontology to compose the whole query condition $C = \{C_1, C_2, ..., C_n\}$.

(2) Matching services. The $C = \{C_1, C_2, ..., C_n\}$ is taken as query condition and through it we can rewrite the user query. That is, we change the query from keywords query to ontology query. Finally an available result set is obtained.

The process of the grammar and semantic query stages is as follows:

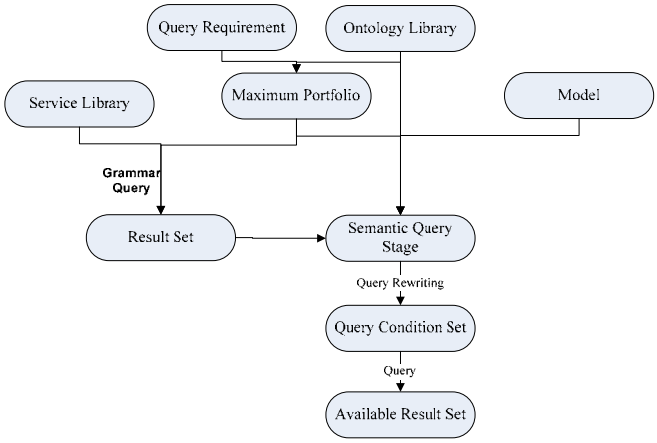


Figure8: grammar semantic query process

In this process, the ideas of ontology and database model are used in the semantic query stage. Its process is that the whole ontology is obtained through the part by means of query. Then we rewrite query and get query condition set. In the end available result set is got.

For service query, firstly we define and describe the service.

[15] Provides uniform definition of services as follows:

{
Primary Information and Provider Information
Functional Description
Quality Description
Other Attributes Description
}

Referring to above four parts, we set down detailed service structure as follows:

{
ServiceName // service name
FunctionDescription// function explanation
Domain// service application domain
ApplyScope //service applicative scope
Input //service input parameters
Output //service output parameters
Precondition // the precondition of service
Postcondition // the post condition of service
Restrictcondition // the restricted condition of service

Namespace // the namespace of WSDL file
Location // the location of WSDL file
}

WSDL [18] (Web Service Description Language Web) is an XML-based language and used to syntactically describe a Web service at the interface and binding levels. At the interface level, abstract interfaces of a Web service are described as interfaces which comprise a set of operations. An operation is in turn defined by its inputs, outputs and fault messages. XML Schema language is used to describe the content of those messages. At the binding level, the service's abstract interface is bound to a particular transport protocol defining specific implementation information such as encoding format and address information [1]. WSDL documents are semi-structured data that describe the functional and non-functional semantics of services [2].

Service description structure with WSDL is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://www.zzl.org/Sum"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.zzl.org/Sum">
<wsdl:types>
<xsd:schema
targetNamespace="http://www.zzl.org/Sum">
<xsd:element name=" MatchRule">
<xsd:complexType>
<xsd:sequence>
<xsd:element name=" ServiceName " type="xsd: string "
/>
<xsd:element name=" FunctionDescription "
type="xsd:string" />
<xsd:element name=" Domain " type="xsd:string" />
<xsd:element name=" ApplyScope " type="xsd:string" />
<xsd:element name=" Input" type="xsd:string" />
<xsd:element name=" Output" type="xsd:string" />
<xsd:element name=" Precondition" type="xsd:string" />
<xsd:element name=" Postcondition" type="xsd:string"
/>
<xsd:element name=" Restrictcondition"
type="xsd:string" />
<xsd:element name=" Namespace" type="xsd:string" />
<xsd:element name=" Location" type="xsd:string" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
</wsdl:types>
</wsdl:definitions>
```

An example is shown to explain the algorithm process.

For example: there exists service ontology Service_Ontology(ServiceID, ServiceName, FunctionDescription, Domain, Input, Output)， in which ServiceID is primary key.

Service relation view Service_view(ServiceID, ServiceName, Domain, ApplyScope, Precondition, Postcondition, Restrictcondition, Namespace, Location)

It links with Service_Ontology by ServiceID.

User query input: Q: FunctionDescription ： startConference, Domain: Multimedia, Input: s_id

User wants to get services of realizing starting conference in multimedia field from Service_view.

The steps of this algorithm are as follows:

Grammar query stage:

Firstly, from the three conditions the maximum portfolio need be obtained. Judging all the input parameters in the ontology library, none is obtained. That is, the user's input parameters are not suitable and we need to generate the maximum portfolio by the second approach of maximum portfolio. The number of elements of composition is two and the result is shown as follow:

Table1: maximum portfolio and granularity

NO	Composition	Availability	Granularity
1	C_1, C_2	Yes	2
2	C_1, C_3	No	1
3	C_2, C_3	No	1

From the above table, the NO1, which is C_1 and C_2 , is available. Therefore, it is not necessary to judge the case in which the number of elements of composition is one. The maximum portfolio is C_1 and C_2 . We take the pretreatment of Service_view. The process is as follow.

Query condition is:

C_1：FunctionDescription like startConference，C_2 ：Domain =Multimedia

Using logic *or* relation between conditions, namely, C_1 *or* C_2 , query result set gets rid of independent records. Grammar query available result set S_1 from Service_view is obtained.

Semantic query stage:

By means of condition C_1 and C_2 , using logic *and* relation between conditions, the query is done in Service_Ontology. Through parts of ontology we can get other parts data. If ontology records are obtained, query condition becomes C set.

So we change query from the keywords query to ontology query. Because that operation to ontology changes into operation to database and ServiceID is primary key, we can use the value of ServiceID in S_1 to get the available result set.

Algorithm analysis:

(1) As the service matching algorithm in UDDI is based on key words matching, matching operation in UDDI matching algorithm is similar with operation in grammar

query stage in the proposed approach in this paper. Namely, this algorithm includes service matching algorithm in UDDI. From our earlier work, performance of the algorithm is nicer than the one of algorithm in UDDI.

In semantic query stage, by means of ontology information, the related data of ontology are obtained. So we can get available result set more accurately.

(2) Because many factors need to be considered in the definition of Web service, such as function description, availability and so on. The interface description of Web service is sophisticated. In the description information the importance of each item is different. Just considering it, we classify description items of Web service and assign different weight. The definition of matching degree is as follow:

$$d(s_i, s_j) = \frac{\sum P_k \times V_k}{|s_i|} \times 100\% \quad . \tag{2}$$

Where, s_i is the Web service which will be matched. s_j is the Web service which will match with s_i. V_k is the weight of the k part of s_i and s_j. The principle of distributing weight is that the weight of basic information is high and the other is low. P_k is the number of items which are consistent in the s_i and s_j. The value is P_k shown as follow:

$$P_k = \sum_{n=1}^{m} E_n \quad . \tag{3}$$

$$E_n = \begin{cases} 1 & s_{i_n} = s_{j_n} \\ 0 & s_{i_n} \neq s_{j_n} \end{cases} . \tag{4}$$

From the matching process of this algorithm, we can see that matching object transforms from key-word matching query to ontology matching query. Because the user is not familiar with the Web service, the matching information is inaccurate. If key-word is used to query to obtain the available result set, matching degree is very low. However if ontology is used to query, because ontology consist of detailed information which are correlated with Web service, the matching degree is very high, even is 100% which indicates matching result set is available and satisfying.

(3) In the process of obtaining the maximum portfolio, the cycle number is n and in every cycle the operation number is C_n^i which is obtained from permutation and combination theory. So the time complexity is $O(n^2)$. In the service matching process, the time complexity is $O(n)$ .So the time complexity of the improved service matching algorithm is

$$O(n^2 + n) = O(n^2) \quad . \tag{5}$$

(4) In the algorithm by query rewriting theory, query changes from keywords to ontology. At the same time, query granularity becomes big. Before, we search through keywords. The keywords are indivisible and minimum granularity. This paper provides an approach by means of ontology and the query granularity largens from keywords to ontology. The ontology operation is more convenient than keywords.

IV. Experiment and Analysis

This algorithm is implemented in the Service Generating Platform (SGP) . SGP is on the base of the architecture of the MVC (Model-View-Control) and refer to the theory of reusing the whole process [4]. In SGP, the language of BPEL (Business Process Execution Language) is used. BPEL has been established as an OASIS (Organization for the Advancement of Structured Information Standards) [17] standard for modeling executable and abstract business processes by orchestrating Web services [16]. Modeling business processes in general and modeling BPEL processes in specific could be enormous time-consuming and error-prone. Reuse has been proved as a valuable approach to avoid reinventing the wheel, take the burden of repeated work off users and improve the quality and efficiency of process modeling.

The platform transforms the abstract process to BPEL-coded process. In the platform we can extract logical processes according to concrete businesses and compose the high-level business process. Therefore, it takes the burden of concrete knowledge of BPEL language off business developers. They only consider the business familiar to them. The aim of the platform decreases developer's burden and increases efficiency of developing SOA applications.

The MutliMedia system is an application of the platform. The data of MutliMedia is stored in the database. The Web services which are stored in the database are minimum granularity. On the base of Web services we can composite models which are divided into two types: Collaborating Model (CM) and Collaborating Model Example (CME) . The different is CM and CME is that the CM is not executable in real-world because only logic structure is preservated and the CME is available in real-world because executable Web services are bound. Their relation graph is as follow:

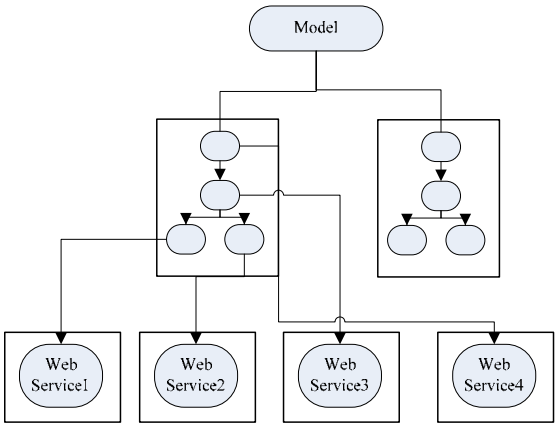


Figure9: CM and CME in Model

In which, the left is CME and the right is CM.

In SGP, we generate a new business process which may be different granularity by CM, CME and Web service. In this case, we can improve development efficiency and save time for developers.

The process of generating models is shown as figure 10. In a segment program with complete structure, four Web services are correlated with the Logic Object. Getting rid of the four Web services, the logic object is remainder. In Logic Object, the logic relation and cited position of Web services are preserved with some variables. This is, except the fours Web services, the remainder is preserved. The Logic Object is called as CM.

The matching process is the inverse process of the generating process. Its intent is that from the database some suitable candidates can be found out to cover for the Web services which are correlated with the Logic Object originally. The process is that from the user's input parameter set we choose the maximum portfolio and using it in the ontology library obtain the ontology which is coincident with user's requirement and get the available result set through querying database.

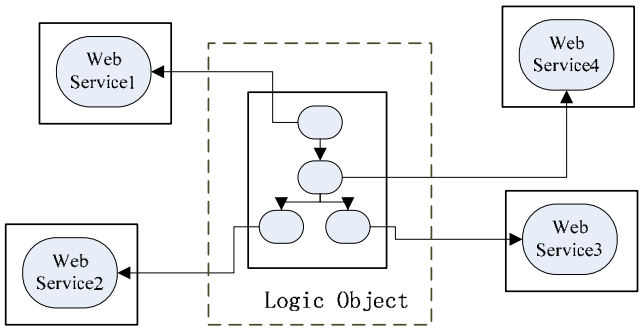


Figure10: Process of Generating CM in CEWS

The run-time environment is that CPU is double kernels, 1.8G; memory is 2GB; operation system is WindowsXP; database is MySQL. Data model of database is consistent with the above the structure of example. All the services of multimedia video conference are stored in database, in which every record represents a service. There are four query requirements. The query condition is as follows:

Table2: Query condition

NO.	Query condition
1	ServiceName LIKE '%conference%'
2	ServiceName LIKE '%conference%', Domain = 'videomedio', FunctionDescription LIKE '% conference %'
3	ServiceName LIKE '%conference%', Domain = 'multimedio', FunctionDescription LIKE '%start conference%'
4	ServiceName LIKE '%conferernce%', Domain= ' videomedia ', FunctionDescription LIKE '% conference %', Precondition = 'none'

The Maximum portfolio is obtained as follows:

Table3: Maximum portfolio

NO.	Maximum portfolio	Portfolio granularity
1	ServiceName LIKE '%conference%'	1
2	ServiceName LIKE '%conference%', FunctionDescription LIKE '% conference %'	2
3	ServiceName LIKE '%conference%', Domain = 'multimedio',	2
4	FunctionDescription LIKE '% conference %', Precondition= 'none'	2

Query result is:

Table4: Query Result

NO.	SNG	SNS	Query Accurate Rate
1	2	2	50%
2	10	2	50%
3	30	2	50%
4	13	4	100%

SNG indicates service number which is obtained in grammar query stage.

SNS indicates service number which is obtained in semantic query stage.

Note: the number of services which completely satisfy user requests is four. The Query Accurate Rate is obtained by the under formula shown as follow:

$$QAR = SNS / SN \times 100\% \quad (6)$$

In which, SN indicates satisfying service number which is four here.

From the data, we can see that maximum portfolio algorithm comes into force. When the number of query condition is one (query condition 1) , although maximum portfolio algorithm is used, there is not any effect. Even though there is misspelling, the algorithm is helpless. In the Table2 and Table3, query condition 2 indicates that in ontology library, condition ServiceName LIKE '%conference%' and FunctionDescription LIKE '% conference %' are maximum portfolio and the PG is 2.

The maximum portfolio algorithm plays an important role. The reason is that condition ServiceName LIKE '%conference%' and FunctionDescription LIKE '% conference %' are in one domain and condition Domain = 'videomedia' is in another domain. Condtion3 represents that in keyword array there is mistake in condition FunctionDescription LIKE '%start conference%'. So PG is 2. The condition 4 indicates there are mistakes of condition2 and condition3. So its PG is 2.

From the Table4, we can see that although sometimes query result set is not better, it is very helpful for the user. User may think about query condition again based on the result set and use accurate condition to obtain satisfying result set.

V. CONCLUSION

This paper provides an available approach to deal with Web service discovery problem. The aim of maximum portfolio is that suitable maximum portfolios are found out from ambiguous input keyword set. We can obtain the best query result set for one time input. So from the principle of the maximum portfolio we can see that it can decrease user's input time and find out suitable compositions from the ambiguous input keyword set. The approach is applied to the SGP and experiments show the algorithm is feasible and effective.

With the rapid development of the Web service technology, we will confront more and more new problems and numerous challenges of Web service technology. Further research about other approach to solve the issue is necessary. We promote the technological development in the process of solving problems about engineering techniques. But there are some problems to solve. For example in this algorithm, the searching method of maximum portfolio are complicated, Better approaches will be created for this question. In addition, the algorithm performance needs to be verified based on huge data. In the future we will collect more data to verify the algorithm performance.

ACKNOWLEDGEMENT

This research is supported by the National 973 Programs (Grant No. 2011CB302704) . National Natural Science Foundation of China (Grant No. 61003067) , Chinese Universities Scientific Fund (Grant No. 2009RC0507) and Key Projects for Science and Technology Development (No. 2011ZX03002-002-01) .

REFERENCES

[1] Tran Vuong Xuan, Puntheeranurak, Hidekazu TSUJI. A new service matching definition and algorithm with SAWSDL.2009 3rd IEEE International Conference on Digital Ecosystems and Technologies, DEST '09, pp.371-376

[2] Qianhui Althea Liang, Herman Lam. Web Service Matching by Ontology Instance Categorization. Proceedings of the 2008 IEEE International Conference on Services Computing, pp. 202 – 209

[3] Hongen Lu. Semantic Web services discovery and ranking. In proceedings of 2005 IEEE/WIC/ACM International Conference on Web Intelligence, pp.157 – 160

[4] Budan Wu, Zhi Jin, Bin Zhao. A Modeling Approach for Service-Oriented Application Based on Extensive Reuse. Proceedings of IEEE International Conference on Web Services (ICWS2008) , Beijing, 2008, pp.754-757

[5] PHui Guo, Anca Andreea Ivan, Rama Akkiraju, PRichard Goodwin. Learning ontologies to improve the quality of automatic web service matching. May. 2007 Proceedings of the 16th international conference on World Wide Web, pp.1241-1242

[6] Unai Aguilera, Joseba Abaitua, Josuka Díaz, David Buján, Diego López de Ipiña. A Semantic Matching Algorithm for Discovery in UDDI .Semantic Computing , 2007. ICSC 2007 .International Conference on17-19 Sept. 2007 pp.751 – 758

[7] Zhi Yang, Junliang Chen, Budan Wu.A New Ontology-based Service Matching Algorithm.2010 IEEE Congress on Services,2010

[8] Sanchez, Christian, Sheremetov, Leonid. A model for semantic service matching with leftover and missing information. Proceedings - 8th International Conference on Hybrid Intelligent Systems, HIS 2008, pp.198-203

[9] Hai Wang, Zengzhi Li. A Semantic Matchmaking Method of Web Services Based on SHOIN^+ (D) *. IEEE Asia-Pacific Conference on Services Computing, APSCC '06, 2006

[10] Pottinger R, Halevy A. MiniCon: a scalable algorithm for answering queries using views. The Int'l Journal on Very Large Data Bases, 2001, 10(2-3) ,pp.182-198

[11] Lin Yue, Song Bao-hua, Duan Hai-bo, Huang Feng-lei. Overview of Semantic Technology and App lications .ComputerApplicationResearch.2005/06

[12] Deng Zhihong, Tang Shiwei, Zhang Ming Yang Dongqing, Chen Jie. Overview of Ontology. Acta Scicentiarum Naturalum Universitis Pekinesis , 2002, 38 (9) ,pp.728- 730

[13] Studer R, Benjamins V R, Fensel D. Knowledge Engineering: Principles and Methods. Data and Knowledge Engineering ,1998 ,25 (122) :161～197

[14] Li Jie. Research On on OWL Ontology Storage Mode. China Science and Technology Information. Nov.2007

[15] J. Fan, B. Ren, L. Xiong. An Approach to Web Service Discovery Based on the Semantics. Proceedings of 2005 International Conference on Fuzzy Systems and Knowledge Discovery, 2005, pp. 1103 – 1106

[16] Zhilei Ma, Frank Leymann. BPEL Fragments for Modularized Reuse in Modeling BPEL Processes. Networking and Services, 2009. ICNS '09. Fifth International Conference on,pp: 63 – 68

[17] OAS IS ,http://www.oasis-open.org/who/

[18] D. Booth and C. K. Liu, eds. Web Services Description Language (WSDL) 2.0, 2006.http://w3.org/TR/2006/CR-wsdl20-primer20060327/

Zhi Yang was born in 1977 and received the M.S. degrees from the School of Computer Science and Technology, North China Electric Power University in 2008. He is currently working toward the Ph.D. degree at State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and

Telecommunications, China. His research interests include service description and discovery, ontology creating and reasoning. Zhi Yang is a student member of ACM and CCF.

Budan Wu received her Ph.D. (2009) degree on Computer Software and theory from Academy of Mathematics and Systems Science, Chinese Academy of Sciences. She joined Beijing University of Posts and Telecommunications in 2009, where she is now Assistant Professor in computer applications. Her research interests include SOA modeling, service-oriented software engineering, requirements engineering and business process modeling. She has published over 15 papers in various conferences and in journals such as ICWS, SCC, COMPSAC, and Chinese Journal of Computers. Budan Wu is a member of ACM and CCF. She is also a secretary for Technical Committee on Services Computing of CCF.

Jun-Liang Chen was born in 1933. He received his associate Ph.D. from Moscow Institute of Telecommunications Engineering in 1961. Now he is a professor at Beijing University of Posts and Telecommunications, China. He is an academician of both Chinese Academy of Sciences and Chinese Academy of Engineering. His current research interests include communication network and communication software, service computing and Web service.

Pingli Gu was born in 1980, and received the M.S. degrees from the School of Computer and Communication, China University Of Petroleum in 2009. She is currently working toward the Ph.D. degree at State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China. Her research interests include service platform and cloud computing.

ID-Based Sequential Aggregate Signatures

Xiangguo Cheng
School of Information Engineering, Qingdao University, Qingdao 266071, China
Email: chengxg@qdu.edu.cn

Lifeng Guo, Chen Yang and Jia Yu
School of Computer and Information Technology, Shanxi University, Taiyuan 030006, China
China Electronics Standardization Institute, Beijing 100007,China.
School of Information Engineering, Qingdao University, Qingdao 266071, China
Email: lfguo@sxu.edu.cn yangchenyf@163.com yujia@qdu.edu.cn

***Abstract*—An aggregate signature provides a method for combining *n* signatures of *n* different messages from *n* different signers into one signature of unit length. The main benefit of such schemes is that they allow bandwidth and computational savings. There exist several trials for the construction of ID-based aggregate signature schemes so far. Unfortunately, the computational complexity and (or) signature length of these schemes grow linearly with the number of signers. This paper focuses on the solution of these problems and proposes a new ID-based sequential aggregate signature scheme based on IB-mRSA. It is compatible with RSA and has the fixed signature length. The security analysis shows that it is secure in the random oracle model with the assumption of classical RSA.**

***Index Terms*—ID-based cryptography, digital signature, aggregate signature, RSA**

I. INTRODUCTION

An aggregate signature, primitively proposed by Boneh *et al.*[2], is a signature that support aggregation: Given *n* signatures on *n* distinct messages from *n* distinct users, it is possible to aggregate all these signatures into one signature. Such a signature (and all the original messages) will convince any verifier that the *n* users did signed the *n* original messages. Thus an aggregate signature provides non-repudiation at once on many different messages by many users. Aggregate signatures are useful for reducing the size of certificate chains (by aggregating all signatures in the chain) and for reducing message size in secure routing protocols such as SBGP[1], *etc*. However, such aggregation technique is only used by the verifier after all the individual signatures having been finished, while the individual signatures have to be sent along with the signed messages. This is restricted in the environment of low bandwidth and storage. Lysyanskaya *et al*. proposed a sequential aggregate signature scheme from trapdoor permutations [3]. A sequential aggregate signature is in fact an aggregate signature that signature aggregation can be done during the signing process. Each signer in turn sequentially adds his signature to the current aggregation. Aggregation of the individual signatures is performed incrementally and sequentially. In a sequential aggregate signature scheme, signing and aggregation are finished at the same time and only the aggregated signature is to be sent to the next signer.

The concept of identity (ID)-based cryptography was first introduced by Shamir [5] in 1984. Its aim is to eliminate the need for public key certificates by allowing a public key to be uniquely derived from a user's identity information. ID-based public key setting can be a good alternative for certificate-based public key setting, especially when efficient key management and moderate security are required. Many ID-based encryption and signature schemes [6-13] have been proposed since 1984. But none of them provides efficient solutions to revoke a user's identity. Boneh *et al*. [14] proposed an ID-based mediated RSA (IB-mRSA) scheme. IB-mRSA is a practical and RSA [4] compatible method of splitting the private key corresponding to a user's ID between the user and the security mediator (SEM). Neither the user nor the SEM knows the factorization of the RSA modulus and neither can sign/decrypt message without the other's help. IB-mRSA not only presents a practical ID-based cryptography, but also provides an efficient solution to the fast revocation of a user's ID.

There exist several trials for constructing ID-based aggregate signature schemes [15-20] so far. However, the schemes are not quite ID-based aggregate signature schemes in the original sense of [2] since they require an additional communication round to aggregate random parts of ID-based signatures provided by multiple signers into a single element [15], or a certain synchronization for sharing the same random string [20], or their signature length grows linearly with the number of signers [16-19]. In fact, the most difficult issue in the construction of such a scheme is how to reduce the aggregate signature length from $O(n)$ to $O(1)$ for n signers. This paper focuses on the solution of this issue. To see the difficulty, we note that almost all the previous ID-based aggregate signature schemes are constructed from some ID-based signature schemes based on bilinear pairing, and these schemes, unlike BLS signature scheme [21], are not deterministic. If each successive signer contributed a randomness to the

Manuscript received Jan 20, 2011; revised Mar 30, 2011; accepted Apr 5, 2011.
Corresponding author: Xiangguo Cheng

doi:10.4304/jsw.6.12.2495-2499

aggregate signature in a trivial way, this randomness would cause the size of the signature to grow linearly with n. IB-mRSA is a deterministic ID-based signature scheme. It allows multiple users of the system to share the same modulus, and each private-public key pair corresponding to an ID is generated by a trusted Private Key Generator (PKG), not by the signer himself, which guarantees each trapdoor permutation is a certified one. These properties of IB-mRSA provide us a good way to realize the construction of an ID-based sequential aggregate signature scheme using the method given in [3].

II. IB-mRSA Signature Scheme

The main idea behind IB-mRSA is to introduce a security mediator (SEM) in classical RSA. The private key corresponding to a user's ID is divided into two parts by PKG. One part is given to the user and the other is given to SEM. Neither the user nor SEM can sign/decrypt a message without the other's help. As a result, a user's ID (*i.e.* sign/decrypt capability) can be immediately revoked by asking SEM not to help him any more. To prepare for our scheme, we first give a review of IB-mRSA as follows.

Setup: Given a security parameter κ, PKG randomly chooses a κ-bit RSA modulus $n = pq$, where p and q are two $\kappa/2$-bit primes. Define two hash functions:

$$H_1 : \{0,1\}^* \to \{0,1\}^l,\ H_2 : \{0,1\}^* \to \mathbb{Z}_n$$

where l is a parameter depending on κ. PKG broadcasts n, H_1, H_2.

Extract: Given a user's identity ID, PKG computes $e = 0^s \| H_1(ID) \| 1$ and $d = e^{-1} \bmod \varphi(n)$, where $s = k - l - 1$. Then it chooses a number $d_u \in_R \mathbb{Z}_n^*$ and computes $d_s = (d - d_u) \bmod \varphi(n)$. d_u and (d_s, *ID*) are secretly sent to the user and SEM, respectively.

Sign: To sign a message *m*, the signer sends *m* along with his ID to SEM. They perform the following tasks in parallel.

—SEM first checks that the signer's ID has not been revoked. It then computes $Sig_s = H_2(m)^{d_s} \bmod n$ and sends it back to the signer.

—The signer computes $Sig_u = H_2(m)^{d_u} \bmod n$ and $\sigma = (Sig_s \cdot Sig_u) \bmod n$. It checks whether $H_2(m) = \sigma^e \bmod n$. If so, the signature on *m* under ID is set to be σ.

Verify: Given a signature σ of message *m* under ID, the verifier computes $e = 0^s \| H_1(ID) \| 1$. He accepts the signature if $H_2(m) = \sigma^e \bmod n$.

Theorem 1. The IB-mRSA signature scheme is unforgeable in the random oracle model under the assumption of classical RSA.

Proof: Note that the IB-mRSA signature is in fact a (2,2) threshold signature. A threshold signature scheme is unforgeable if the underlying signature scheme is secure and the threshold signature is simulatable [17]. The underlying signature is a classical RSA signature. In the following, we need only to show that the IB-mRSA signature is simulatable.

To prove the simulatability of the IB-mRSA signature scheme, we construct a simulator *SIM* to simulate the IB-mRSA signature generation protocol *Sign*. Suppose that an adversary *A* has corrupted a signer whose identity is ID. His goal is to forge a signature of this signer without the help of SEM. The view of an adversary *A* consists of the message *m*, the modulus *n*, the signer's public-private key pair (e, d_u), and the signature σ of *m* under ID. Let $VIEW_A(Sign(n, e, d_u, m), \sigma)$ denote all the information that *A* is able to get. *SIM*'s inputs are the message *m*, the modulus *n*, the signer's public-private key pair (e, d_u), and the signature σ. Let $SIM(n, e, d_u, m, \sigma)$ denote all the information produced by the simulator. The following description shows that $VIEW_A(Sign(n, e, d_u, m), \sigma)$ is computationally indistinguishable from $SIM(n, e, d_u, m, \sigma)$.

On the one hand, the partial signatures given by the signer and SEM are

$$Sig_u = H_2(m)^{d_u} \bmod n,\ Sig_s = H_2(m)^{d_s} \bmod n.$$

Both are random numbers in $\mathbb{Z}_n$ since d_u is randomly chosen from $\mathbb{Z}_n$ and $d_s = (d - d_u) \bmod \varphi(n)$; On the other hand, the partial signature of the corrupted user in *SIM* can be computed as $Sig'_u = H_2(m)^{d_u} \bmod n$, and the partial signature of the SEM in *SIM* is Sig'_s such that $\sigma = (Sig'_s \cdot Sig'_u) \bmod n$. They are also random numbers in $\mathbb{Z}_n$. Therefore, Sig_u, Sig_s, Sig'_u and Sig'_s have the same distribution in $\mathbb{Z}_n$.

We recall that it is completely insecure to have a common modulus for several users in classical RSA since the knowledge of a single private-public key pair allows a user to factor the modulus. However, IB-mRSA allows multiple users to share the same modulus since neither the users nor SEM is able to completely know the private key corresponding to an ID. Note that collusion between a user and SEM would result in a total break of the whole scheme. Therefore, SEM here must be assumed to be a totally trusted and secure entity and no user is able to compromise it.

III. ID-Based Sequential Aggregate Signature

We introduce the definition of an ID-based sequential aggregate signature and present its security model in this section.

A. Definition of ID-Based Sequential Aggregate Signature

An ID-based sequential aggregate signature can be viewed as a combination of a sequential aggregate signature and an ID-based signature. Namely, it is a sequential aggregate signature, but, all the public keys are the users' IDs. It generally consists of four algorithms: *Setup*, *Extract*, *Aggregate Signing* and *Aggregate Verify*.

Setup: Given a security parameter κ, PKG generates and publishes the system parameters.

Extract: Given an identity ID_i of a user U_i, PKG generates a private key sk_i corresponding to ID_i and secretly sends it to U_i.

Aggregate Signing: Signing and aggregation is a combined operation. The operation takes as inputs a private key sk_i, a message m_i to sign, and a sequential aggregate signature σ' on messages $m_1, m_2, \cdots, m_{i-1}$ under $ID_1, ID_2, \cdots, ID_{i-1}$, where m_1 is the inmost message. It adds a signature on m_i under ID_i and outputs a sequential aggregate signature σ on all messages $m_1, m_2, \cdots, m_i$.

Aggregate Verify: Verifies that σ is a valid sequential aggregate signature on messages $m_1, m_2, \cdots, m_i$ under $ID_1, ID_2, \cdots, ID_i$.

B. Security of ID-Based Sequential Aggregate Signature

The security of an ID-based sequential aggregate signature scheme is defined as the non-existence of an adversary capable of existentially forging an ID-based sequential aggregate signature. Existential forgery here means that the adversary attempts to forge a sequential aggregate signature, on some messages of his choice, under some set of IDs.

Similar to [3], we formalize the security model of an ID-based sequential aggregate signature scheme as sequential aggregate chosen ID security. In this model, the adversary *A* is first given an ID. His goal is the existential forgery of an ID-based sequential aggregate signature. We give *A* the power of choosing all IDs expect the challenge ID. The adversary is also given access to a sequential aggregate signing oracle on the challenge ID. We say the adversary succeeded if he won the following game.

Setup. The adversary *A* is provided with an ID, generated at random.

Queries. Proceeding adaptively, *A* can request the public-private key pairs corresponding to some IDs except the provided ID. *A* can also request the sequential aggregate signatures under these IDs on messages of his adaptive choice. For each query, we allow *A* to supply a sequential aggregate signature σ' on some messages $m_1, m_2, \cdots, m_{i-1}$ under some distinct identities $ID_1, ID_2, \cdots, ID_{i-1}$ and an additional message m_i to be signed under ID.

Responses. Finally, *A* outputs *j* distinct identities $ID_1, ID_2, \cdots, ID_j$, *j* messages $m_1, m_2, \cdots, m_j$ and a sequential aggregate signature σ.

A wins the game if σ is a valid sequential aggregate signature on messages $m_1, m_2, \cdots, m_j$ under some distinct identities $ID_1, ID_2, \cdots, ID_j$, and one of these identities is ID.

IV. ID-Based Sequential Aggregate Signature Scheme

Using the IB-mRSA signature scheme as an underlying scheme, we first manage to construct an ID-based sequential aggregate signature scheme, then we show the security analysis of this scheme. For the convenience of description, we first introduce some notations used in our scheme. Suppose that (*e*, *d*) is a public-private key pair of RSA, then $\pi(x) = x^e \bmod n$ is a permutation on $\mathbb{Z}_n^*$ and $\pi^{-1}(x) = x^d \bmod n$ is its inverse, which are derived uniquely from *e* and *d*. In the following, we use (π, π^{-1}) to denote (*e*, *d*) and use $a +_n b$ to denote the operation of $(a+b) \bmod n$.

A. Proposed Scheme

Our ID-based sequential aggregate signature scheme is described as follows:

Setup: It is the same as that in the underlying IB-mRSA signature scheme.

Extract: Given an identity ID_i of user U_i, PKG computes $e_i = 0^s \| H_1(ID_i) \| 1$ and $d_i = e_i^{-1} \bmod \varphi(n)$, where $s = k - l - 1$. Then it chooses a number $d_i^u \in_R \mathbb{Z}_n^*$ and computes $d_i^s = (d_i - d_i^u) \bmod \varphi(n)$. d_i^u is sent to U_i and (d_i^s, ID_i) is sent to SEM.

Aggregate Signing: Without lose of generality, suppose that *j* signers $U_1, U_2, \cdots, U_j$ orderly generate an ID-based sequential aggregate signature on *j* messages $m_1, m_2, \cdots, m_j$. Signer U_1 first signs message m_1. He interacts with SEM to do the following work:

— U_1 computes $h_1 = H_2(ID_1, m_1)$ and sends it to SEM.

— SEM first checks that U_1's identity ID_1 has not been revoked. If so, it computes a partial signature $Sig_1^s = h_1^{d_1^s} \bmod n$ and sends it back to U_1.

— U_1 computes $Sig_1^u = h_1^{d_1^u} \bmod n$. After doing this, he then checks whether $h_1 = (Sig_1^s \cdot Sig_1^u)^{e_1} \bmod n$ holds. If so, the signature on message m_1 under ID_1 is set to be $\sigma_1 = (Sig_1^s \cdot Sig_1^u) \bmod n$ and is sent to the second user U_2.

Using the above π notation, σ_1 can be written as $\sigma_1 = \pi_1^{-1}(h_1)$.

Having received σ_1, U_2 first verifies that σ_1 is a valid signature on m_1 under ID_1 using the verification algorithm *Verify* of IB-RSA. Suppose that σ_1 is valid, he then computes $h_2 = H_2(ID_1 \| ID_2, m_1 \| m_2)$ and $h_2' = h_2 +_n \sigma_1$. After a procedure of interacting with SEM, U_2 can obtain a signature $\sigma_2 = \pi_2^{-1}(h_2') = \pi_2^{-1}(h_2 +_n \pi_1^{-1}(h_1))$, ……. For the *j*-th signer U_j, having received σ_{j-1}, he first verifies its validity using the algorithm *Aggregate Verify*. If so, U_j computes $h_j = H_2(ID_1 \| ID_2 \| \cdots \| ID_j, m_1 \| m_2 \| \cdots \| m_j)$ and $h_j' = h_j +_n \sigma_{j-1}$. With the help of SEM, U_j can obtain his signature

$$\sigma_j = \pi_j^{-1}(h_j') = \pi_j^{-1}(h_j +_n \pi_{j-1}^{-1}(h_{j-1} +_n \pi_{j-2}^{-1}(\cdots \pi_2^{-1}(h_2 +_n \pi_1^{-1}(h_1)))))$$

where $h_i = H_2(ID_1 \| ID_2 \| \cdots \| ID_i, m_1 \| m_2 \| \cdots \| m_i)$ for $i = 1, 2, \cdots, j$. The ID-based sequential aggregate signature on messages $m_1, m_2, \cdots, m_j$ under $ID_1, ID_2, \cdots, ID_j$ is set to be $\sigma = \sigma_j$.

Aggregate Verify: Given the ID-based sequential aggregate signature σ on messages $m_1, m_2, \cdots, m_j$ under

$ID_1, ID_2, \cdots, ID_j$, the verifier sets $\sigma_{i-1} = \pi_i(\sigma_i) +_n (-h_i)$ for $i = j, j-1, \cdots, 2, 1$, he accepts the sequential aggregate signature if $\sigma_0 = 0$, where 0 is the zero element of $\mathbb{Z}_n$.

B. Security Analysis

Using the security model of ID-based sequential aggregate signature scheme given in Section III, we analyze the security of our scheme.

Theorem 2. The proposed scheme is secure against existential forgery under an adaptive sequential aggregate chosen ID attack in the random oracle.

Proof. Suppose that there is a forger F that breaks the security of our ID-based sequential aggregate signature scheme, we will construct an algorithm F' to forge the based IB-mRSA signature scheme by using F.

F' simulates the challenger and interacts with F as the following.

Setup: F' randomly chooses an identity ID as a chosen ID and sends it to F.

ID Queries: F requests the private-public key pairs corresponding to some identities IDs except the chosen ID. F' makes queries on these IDs to its own oracle and gives the corresponding keys to F.

Hash Queries: F requests a hash on some identities $ID_1, ID_2, \cdots, ID_j$ and some messages $m_1, m_2, \cdots, m_j$. F' makes the same query to its own hash oracle and gives the value back to F.

Aggregate Signature Queries: Proceeding adaptively, F requests an ID-based sequential aggregate signature under the chosen ID on some messages of his choice. For each query, F supplies a sequential aggregate signature σ' on messages $m_1, m_2, \cdots, m_{j-1}$ under distinct identities $ID_1, ID_2, \cdots, ID_{j-1}$ and an additional message m_j to be signed under ID. F' first makes a hash query on $h_i = H_2(ID_1 \| ID_2 \| \cdots \| ID_{j-1} \| ID, m_1 \| m_2 \| \cdots \| m_{j-1} \| m_j)$ and obtains the response h, then F' makes a signature query on $h +_n \sigma'$ and gives the response back to F.

Outputs: Eventually F halts, outputting some messages $m_1, m_2, \cdots, m_j$, some identities $ID_1, ID_2, \cdots, ID_j$, and the corresponding sequential aggregate signature forgery σ. The forgery must be nontrivial: The challenge ID must be in $ID_1, ID_2, \cdots, ID_j$, at some location $i (1 \le i \le j)$, and F must not have asked for a sequential aggregate signature on $m_1, m_2, \cdots, m_i$ under $ID_1, ID_2, \cdots, ID_i$. If F fails to output a valid and nontrivial forgery, F' reports failure and terminates. Otherwise, F' does the following work:

Case 1. The chosen ID is at the end of some identities $ID_1, ID_2, \cdots, ID_j$. That is, $ID = ID_j$. F' requests the hash values of $H_2(ID_1 \| ID_2 \| \cdots \| ID_i, m_1 \| m_2 \| \cdots \| m_i)$ and gets the response h_i for $i = 1, 2, \cdots, j$. Then F' manages to get some signatures from his signing oracle. After having obtained the signature σ_1 of m_1 under ID_1. F' computes $h'_2 = h_2 +_n \sigma_1$ and gets a signature σ_2 on h'_2 under ID_2 from the signature query. The rest may be deduced by an analogy. After having obtain the signature σ_{j-2} on h'_{j-2} under ID_{j-2}, F' computes $h'_{j-1} = h_{j-1} +_n \sigma_{j-2}$. The signature σ_{j-1} on h'_{j-1} under ID_{j-1} can also be obtained from the signature query. We note that the sequential aggregate forgery σ is in fact an IB-mRSA signature of $h'_j = h_j +_n \sigma_{j-1}$ under ID_j, then F' get a forgery by using F.

Case 2. The chosen ID is in the middle of some identities $ID_1, ID_2, \cdots, ID_j$. That is to say, there exists an ID_l, where $1 \le l < j$, such that $ID = ID_l$. F' requests the hash values of $H_2(ID_1 \| ID_2 \| \cdots \| ID_i, m_1 \| m_2 \| \cdots \| m_i)$ and gets the response h_i for $i = l+1, l+2, \cdots, j$. Then F' sets $\sigma = \sigma_j$ and $\sigma_{i-1} = \pi_i(\sigma_i) +_n (-h_i)$ for $i = j, j-1, \cdots, l+1$. At last, F' obtains

$$\sigma_1 = \pi_{l+1}((-h_{l+1}) +_n \pi_{l+2}((-h_{l+2}) +_n \cdots \pi_{j-1}((-h_{j-1}) +_n \pi_j(s_j))))$$

We note that σ_1 is in fact a sequential aggregate signature on messages $m_1, m_2, \cdots, m_l$ under $ID_1, ID_2, \cdots, ID_{l-1}, ID$.

Therefore, Case 2 can be easily derived into Case 1.

If there exists an efficient algorithm F to forger our ID-based sequential aggregate signature scheme, then we can construct an algorithm F', with the same advantage, to forge the underlying IB-mRSA signature scheme. However, Theorem 1 has shown that the IB-mRSA signature scheme is existentially unforgeable. Therefore, Our ID-based sequential aggregate signature scheme is secure against existential forgery.

V. COMPARISON

Compared with previous ID-based aggregate signature schemes, our scheme has some advantages described as follows.

(1) Our scheme is based on IB-mRSA. It is compatible with classical RSA. However, all the previous ID-based aggregate signature schemes are constructed from bilinear pairings. We note that the pairing computation is the most time-consuming in pairing-based cryptosystems. Although there have been many works discussing the complexity of pairings and how to speed up the pairing computation, the computation of the pairing still remains time-consuming.

(2) A "good" aggregate signature scheme should satisfy the following properties:

—Flexibility: This property requires that, any user can add his individual signature into the aggregate signature at any time and without the cooperation of the signers.

—Compactness: Compactness requires that, a series of individual signatures by distinct signers on distinct messages should be compressed into a single, compact aggregate signature.

— Deletion: Deletion requires that, any individual signature can be easily removed from the aggregate signature by the signer.

Schemes [15] and [20] don not satisfy the property of flexibility since they need cooperation of the signers while aggregation. Schemes [16-19] don not satisfy compactness property since the aggregate signature length grows linearly with the number of signers. Our

scheme satisfies the above two properties. All the pairing-based aggregate signature schemes satisfy the deletion property. Unfortunately, our scheme do not satisfies this property since it is sequential, The signers can only inversely remove their individual signatures from the aggregate signature.

An efficiency comparison of our scheme with the existing ones is given in Table I, where √ denotes "satisfy" and × denotes "do not satisfy" and "FP", "CP", "DP" and "SL" are abbreviations of "Flexibility Property", "Compactness Property" , "Deletion Property" and "Signature Length".

TABLE I. FUNCTIONALITY COMPARISON OF THE SCHEMES

Scheme	FP	CP	DP	SL
Scheme[15]	×	√	√	$O(1)$
Scheme[17]	√	×	√	$O(n)$
Scheme[18]	√	×	√	$O(n)$
Scheme[19]	√	×	√	$O(n)$
Scheme[20]	×	√	√	$O(1)$
Our Scheme	√	√	×	$O(1)$

VI. CONCLUSIONS

Based on IB-mRSA signature scheme, we proposed an ID-based sequential aggregate signature scheme and give its security analysis in the random oracle. The advantage of this scheme is that the signature length is the same as the single signature, regardless of the number of signers. Furthermore, it is compatible with classical RSA.

ACKNOWLEDGMENT

This work was supported by a grant from the National Natural Science Foundation of China (No.60703089), the Shandong Province Natural Science Foundation of China (No. ZR2010FQ019) and the Plan Foundation of Science and Technology of Shandong Provincial Education Department of China (No. J08LJ02).

REFERENCES

[1] S. Kent, C. Lynn and K. Seo, "Secure border gateway protocol (Secure-BGP)," *IEEE J. Seclected Areas in Comm.*, Vol. 18(4), 2000, pp. 582-592.

[2] D. Boneh, C. Gentry, B. Lynn and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," *Advances in Eurocrypt'03*, LNCS, Vol. 2656, Springer-Verlag, 2003, pp. 416-432.

[3] A. Lysyanskaya, S. Micali, L. Reyzin and H. Shacham, "Sequential aggregate signatures from trapdoor permutations," *Advances in Eurocrypt'04*, LNCS, Vol. 3027, Springer-Verlag, 2004, pp.74-90.

[4] R. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Commun. ACM*, vol. 21(2), 1978, pp.120-126.

[5] A. Shamir, "Identity-based cryptosystems and signature schemes," *Advances in Crypto'84*, LNCS 196, Springer-Verlag, 1984, pp.47-53.

[6] H. Tanaka, "A realization scheme for the identity-based cryptosystem," *Advances in Crypto'87*, LNCS 293, Springer-Verlag, 1987, pp.341-349.

[7] S. Tsuji and T. Itoh, "An ID-based cryptosystem based on the discrete logarithm problem," *IEEE Journal of Selected Areas in Communications*, Vol. 7(4), pp.467-473, 1989.

[8] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," *Advances in Crypto'86*, LNCS, Vol. 263, Springer-Verlag, 1987, pp.186-194.

[9] U. Feige, A. Fiat and A. Shamir, "Zero-knowledge proofs of identity," *J. Cryptology*, Vol.1, pp.77-94, 1988.

[10] D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing," *Advances in Crypto'01*, LNCS Vol. 2139, Springer-Verlag, 2001, pp.213-229.

[11] X. Yi, "An identity-based signature scheme from the Weil pairing," *IEEE Communications Letters*, Vol. 7(2), 2003, pp.76-78.

[12] J. C. Cha and J. H. Cheon, "An identity-based signature from gap Diffie-Hellman groups," *Advances in Public Key Cryptography-PKC 2003*, LNCS, Vol. 2567, Springer-Verlag, 2003, pp.18-30.

[13] F. Hess, "Efficient identity based signature schemes based on pairings," *Proceedings of Select Areas in Cryptography, SAC 2002*, LNCS, Vol. 2595, Springer-Verlag, 2002, pp.310-324.

[14] X. Ding and G. Tsudik, "Simple identity-based cryptography with mediated RSA," *Proceedings of CT-RSA 2003*, LNCS, Vol. 2612, Springer-Verlag, 2003, pp.193-210.

[15] X. Cheng, J. Liu and X. Wang, "Identity-based aggregate and verifiably encrypted signatures from bilinear pairing," *Proceedings of ICCSA 2005*, LNCS, Vol. 3483, Springer-Verlag, 2005, pp.1046-1054.

[16] J. H. Cheon, Y. Kim and H. J. Yoon, "A new ID-based signature with batch verification," *Cryptology ePrint Archive*, Report 2004/13, available at http://eprint.iacr.org.

[17] H. J. Yoon, J. H. Cheon and Y. Kim, "Batch verification with ID-based signatures," *ICISC'04*, LNCS, Vol. 3506, Springer-Verlag, 2005, pp. 233-248.

[18] J. Xu, Z. Zhang and D. Feng, "ID-based aggregate signatures from bilinear pairings," *CANS'05*, LNCS, Vol. 3810, Springer-Verlag, 2005, pp. 110-119.

[19] J. Herranz, "Deterministic identity-based signatures for partial aggregation," *The Computer Journal*, Vol. 49(3), 2006, pp.322-330.

[20] C. Gentry and Z. Ramzan, "Identity-based aggregate signatures," *Advances in Public Key Cryptography-PKC 2006*, LNCS, Vol. 3958, Springer-Verlag, 2006, pp.257-273.

[21] D. Boneh, B. Lynn, H. Shacham, "Short signatures from the Weil pairing," *Advances in Asiacrypt'01*, LNCS, Vol. 2248, Springer-Verlag, 2002, pp. 514-532.

Unleashing the Potential Impact of Nonessential Self-contained Software Units and Flexible Precedence Relations upon the Value of Software

Antonio Juarez Alencar, Rafael Alcemar do Nascimento, Eber Assis Schmitz,
Alexandre Luis Correa, and Angélica F. S. Dias
Email: juarezalencar@nce.ufrj.br, rafael.alcemar@gmail.com, eber@nce.ufrj.br,
alexcorr@yahoo.com, angelica@nce.ufrj.br

Abstract— **This paper evaluates the impact of nonessential minimum marketable features modules (NMMF) and nonessential architectural elements (NAEs) on software projects; shows that the value-creation path of these self-contained software units may be quite different from that of essential software units; and discusses the impact of early NMMF and NAE identification on the value of software projects to business and the deployment of business strategy. Moreover, the paper demonstrates that the existence of flexible precedence relations among MMFs and AEs may also be exploited to further increase the value of software development initiatives.**

Index Terms— **Value-based software engineering, nonessential software units, minimum marketable feature modules, economics of software engineering.**

I. INTRODUCTION

Despite the vital role that IT can play in the business landscape, funding software development projects in the highly competitive environment in which companies do business these days has become a central issue for both managers and practitioners alike. In many markets stakeholders and investors are claiming not only for better financial returns, but also for shorter investment periods, faster time to market, less risk and improved ability to adapt to new market conditions quickly and efficiently [1], [2]. As a consequence, it is becoming increasingly difficult to get software development projects funded, unless they provide clearly defined value to business [3].

Consistent with this view many attempts have been made to bring financial discipline to software development [4]. While some proposals are broadly based on project financial valuation metrics such as net present value, return on investment and, more recently, real options theory [5], [6], others take a more holistic view of software development and advocate the use of value-based metrics [7]–[9].

Nevertheless, all these attempts fall short of acknowledging that requirement prioritization and modularization play a crucial role in the business value of software. While the requirements satisfied by a module are paramount to determine its value, the order in which these modules are implemented dictates how soon this value can be appropriated.

One notable exception is presented by Dennne and Cleland-Huang [3] who suggest the use of thorough financial analysis to maximize the value of software projects composed of minimum marketable feature modules (MMFs), i.e. modules containing small sets of features that have value to business. In their work Dennne and Cleland-Huang show that the implementation order of MMFs may change quite substantially the value of such projects. The ideas of Dennne and Cleland-Huang were later extended by Alencar *et al.* [10], who use a *branch & bound* technique to overcome some of the limitations imposed by the method proposed by the former [11].

Nonetheless, both Denne and Cleland-Huang, and Alencar *et al.* failed to recognize that (a) it is not always the case that all modules are essential to the development of a software project; (b) if a module is nonessential to a software project, its development may or may not be rightfully pursued by the project manager during the project life cycle; (c) the value of nonessential modules may vary according to the set of modules that have preceded its development and (d) when implemented, instead of creating their own cash flow stream, nonessential modules may contribute to the value of a software just by positively influencing the value of essential modules.

This paper is a step towards filling this gap by uncovering the value of nonessential minimum marketable features modules (NMMFs) and nonessential architectural elements (NAEs), whose value-creation path may be quite different from those of essential self-contained software units. Moreover, this paper shows how the early identification of NMMFs and NAEs during the project life cycle may affect the final value of software projets, benefit software development as a whole and help to shape business strategy. In addition, this paper demonstrates how the combination of flexible precedence relations and nonessential self-contained software units may be used to increase the value of software even further.

The remainder of this paper is organized as follows. Section II presents a review of the principal concepts and methods used in this paper. Section III introduces a real-world inspired example that helps in understanding the role played by NMMFs, NAEs and flexible precedence relations in software development. Section IV presents

doi:10.4304/jsw.6.12.2500-2507

the conclusions of this paper.

II. Minimum Marketable Features

According to Denne and Cleland-Huang [3] MMFs are self-contained software units that create value to business in one or several of the following areas: (a) competitive differentiation (b) revenue generation (c) cost savings (d) brand projection and (e) enhanced customer loyalty. Although an MMF is a self-contained unit, it is often the case that it can only be developed after other project parts have been completed. These project parts may be either other MMFs or the architectural infrastructure, i.e. the set of basic features that offers no direct value to customers, but that are required by the MMFs.

The architectural infrastructure itself can usually be divided into self-contained deliverable elements. These elements, called *architectural elements*, or AEs for short, enable the architecture to be delivered according to demand, further reducing the initial investment needed to run a project. Moreover, the total value brought to a business by a software consisting of several interdependent MMFs and AEs, each one with its own cash flow stream and precedence restrictions, is highly dependent on the development order of these units.

For instance, Figure 1 presents the precedence diagram of self-contained software units comprising a loan control system. The meaning of each software unit is described in Table I. In this particular example SU_1, $\cdots$, SU_5 are MMFs and SU_6 is an AE. The reasons why these particular software units are MMFs and AEs is an object of discussion in Section III.

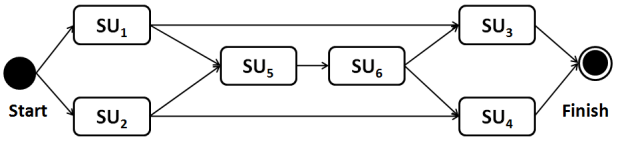


Figure 1. Loan control software precedence diagram.

In the diagram an arrow going from one MMF or AE to another, e.g. $SU_5 \rightarrow SU_6$, indicates that the development of the former (SU_5) must precede the development of the latter (SU_6).

Table II indicates all possible development sequences for the loan control software, considering that (a) it takes exactly one period to develop each software unit; (b) only one software unit can be developed per period; (c) the first software unit is developed in period one and (d) there is no delay between the completion of a software unit and the beginning of the development of the next.

Table III shows the undiscounted cash flow elements of each MMF and AE in the model introduced in Figure 1. For example, according to the information presented in Table III, SU_1 requires an initial investment of US $50 thousand. Once its development is completed at the end of the first period, it provides a series of positive returns until the fifteenth period, when the salary-loan software becomes obsolete and is replaced by a new and more advanced tool. Hubbard [12] shows how the cash flow elements of software projects may be properly estimated.

Because it is improper to perform arithmetic operations on monetary values without taking into account an interest rate, in order to compare the business value of different MMFs and the investment required by AEs one has to resort to their discounted cash-flow [13]. Table IV shows the sum of the discounted cash-flow of each software unit in Figure 1, considering a discount rate of 2% per period. Such a sum is the net present value (NPV) of all cash flow elements of the software unit. In order to make understanding easier, the figures presented in Table IV have been rounded to the nearest integer value. The remaining figures presented in this paper follow the same convention.

For instance, according to the information presented in Table IV, if SU_1 is developed in the first period, it yields a NPV of US $1,045 thousand i.e.

$$\frac{-50}{(1+2\%)^1} + \frac{50}{(1+2\%)^2} + \frac{70}{(1+2\%)^3} + \cdots + \frac{50}{(1+2\%)^{15}}$$

On the other hand, if SU_1 is developed in the second period, it yields a NPV of $987 thousand, in the third

TABLE I.
Description of the Software Units Comprising the Loan Control Software

Software Unit		
Id	**Name**	**Description**
SU_1	Apply for a loan	Collects the necessary data to grant a loan to a customer
SU_2	Apply for refinancing	Collects the data needed to grant the refinancing of an existing loan
SU_3	Accept loan conditions	Allows customer to accept or decline the loan conditions proposed by the lender
SU_4	Accept refinancing conditions	Allows customer to accept or decline the refinancing conditions proposed by the lender
SU_5	Quick credit analysis	Figures the likelihood of a customer paying back a loan in accordance with certain installment values and dates
SU_6	Check for funding availability	Checks whether the lender has the necessary funds to grant a loan to a given customer

TABLE II.
Scheduling Options

Scheduling Options	Period					
	1	**2**	**3**	**4**	**5**	**6**
1	SU_1	SU_2	SU_5	SU_6	SU_3	SU_4
2	SU_1	SU_2	SU_5	SU_6	SU_4	SU_3
3	SU_2	SU_1	SU_5	SU_6	SU_3	SU_4
4	SU_2	SU_1	SU_5	SU_6	SU_4	SU_3

TABLE III.
Software-Unit Cash-Flow Elements

Id	Cash Flow Elements (US $1,000)				
	Period				
	1	**2**	**3**	**4 to 14**	**15**
SU_1	-50	50	70	100	50
SU_2	-70	20	28	40	20
SU_3	-30	500	700	1,500	1,000
SU_4	-40	200	280	600	200
SU_5	-80	90	120	180	90
SU_6	-10	0	0	0	0

TABLE IV.
SOFTWARE-UNIT NET-PRESENT VALUE

Id	Net Present Value (US $1,000)					
	Period					
	1	2	3	4	5	6
SU_1	1,045	987	894	802	712	623
SU_2	368	346	309	274	239	204
SU_3	16,001	14,944	13,537	12,157	10,804	9,478
SU_4	6,221	5,950	5,388	4,836	4,296	3,766
SU_5	1,885	1,781	1,613	1,447	1,285	1,126
SU_6	-10	-10	-10	-9	-9	-9

$894 thousand and so on.

Obviously, not all software units can be developed in the first period. The precedence diagram presented in Figure 1 indicates that only SU_1 and SU_2 can be developed at that time. Because in this example each software unit requires exactly one period to be developed, SU_5 cannot be developed until the third period. Furthermore, each particular sequence of software units yield its own NPV. For instance, the sequence

$$SU_1 \rightarrow SU_2 \rightarrow SU_5 \rightarrow SU_6 \rightarrow SU_3 \rightarrow SU_4$$

yields $17,564 thousand, which is the highest NPV among all possible development sequences.

III. AN EXAMPLE

According to Robert Hall (1764-1831), the British clergyman: "The innocence of the intention abates nothing of the mischief of the example". As a result, the discussion presented in this paper is introduced step-by-step with the help of a real-world inspired example regarding the development of a mobile software application[1].

Step 1: Context information

"Salary loans" are low-risk low-interest multi-purpose loans granted to qualified employees of accredited companies, in which payment is made on installments via salary deduction [14]. In this respect consider an international financial institution such as CITIBANK, BARCLAYS, HSBC, ABN AMRO, UBS and many others that make salary loans available to their customers. For the purpose of this paper this organization is named LOANS "R" US, or L_RU.

As soon as an application for a salary loan is received, L_RU figures the likelihood of the applying customer paying back the requested loan in accordance with acceptable installment values and dates. Next, L_RU makes sure that it has enough funds to grant the loan that the customer is asking for (as by law financial institutions cannot lend money above a certain limit, so as to preserve its financial health). Finally, the applying customer is presented with the conditions under which a salary loan may be granted, if any. At this point, they have the option of accepting or declining the loan offer.

[1]One of the authors of this paper has successfully managed a project that is very similar to the one described in this section for a major financial institution in South America, which kindly asked not to have its name disclosed.

Financial institutions that lend money also tend to offer refinancing services for existing loan agreements as a way of relieving the financial pressure on both current and future customers. L_RU is no exception to the rule. As the refinancing operation may involve a loan agreement issued by a third party, this operation is frequently referred to as "debt purchase".

In order to provide an adequate competitive response to recent competition moves into its salary-loan market and further develop its loan business, L_RU has decided to build a new salary-loan mobile web-based software. The company believes that if it acts fast, this software may not only improve its turnover considerably, but also favorably reshape the competitive landscape of the salary-loan business. Figure 1 introduces a model containing the software units that comprise the new salary-loan software.

Step 2: Determining how each software unit may generate revenue

Table I describes the meaning of each software unit introduced in Figure 1. One should note that the first five software units listed in that table generate revenue for L_RU in the following manner:

- *SU_1 and SU_2* - every client applying for a salary-loan is asked whether they agree to receive new product offers from both L_RU and its associated companies from time to time. Willing customers may then be targeted by new marketing campaigns, generating sales opportunities for L_RU and revenue in the form of fees due to the use of L_RU's customer database by the associated companies;
- *SU_3 and SU_4* - once a client accepts a loan or refinancing offer, they generate revenue in the form of due interest and
- *SU_5* - the results of the "Quick credit analysis" software unit are used to enrich the L_RU's customer database. Credit analysis information is particularly important to marketing campaigns that depend upon the financial heath of willing customers, especially those campaigns that allow customers to pay for a product or service on installments. L_RU's associated companies are required to pay a premium fee to access the enriched database.

Therefore SU_1, SU_2, $\cdots$, SU_5 are all minimum marketable features modules, or MMFs. On the other hand, SU_6, "Check for funding availability", also describes a self-contained software unit, which provides an essential service to the intended behavior of the software, but neither L_RU's customers nor associated companies are willing to pay for the service it provides. Therefore, SU_6 describes an architectural element (AE).

Step 3: Planning the software implementation

Although the project team assembled by L_RU to run the salary-loan software project has initially considered adopting the software units described in Table I together with the precedence diagram presented in Figure 1, they soon realized that the behavior of SU_5 does not actually depend on whether the salary loan under consideration is

the result of a new application or a request for refinancing. Therefore, the development of SU_5 may start when the development of either SU_1 or SU_2 is completed, or even when the development of both of them are completed. Note that this new perspective on the dependencies required by SU_5 is actually a real option, and options may quite substantially change the value of software projects [15]. Hence the L_RU's project team decided to amend the precedence diagram presented in Figure 1. Figure 2 shows the updated precedence diagram for the salary-loan software project.

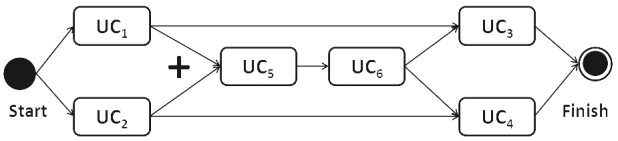


Figure 2. Salary-loan project precedence diagram.

One should note that SU_5's flexible precedence requirement is properly signaled by the presence of the "+" symbol (inclusive or) in the diagram. It is also important to note that the modules presented in Figure 2 comprise the set of modules that are essential to the development of the salary-loan software, as (according to current markets conditions) this is the smallest set of self-contained modules that bears a fair chance of providing long-term lawful return on the investment to be made by L_RU in their development [16]. Table V indicates all possible development sequences for the salary-loan software. There are fourteen possibilities altogether. It is important to mention that as modules SU_1, SU_2, $\cdots$, SU_6 are all essential to the intended behavior of the software, therefore all of them have to be developed eventually.

TABLE V.
SCHEDULING OPTIONS

Scheduling Options	Period					
	1	2	3	4	5	6
1	SU_1	SU_2	SU_5	SU_6	SU_3	SU_4
⋮	⋮	⋮	⋮	⋮	⋮	⋮
7	SU_1	SU_5	SU_6	SU_3	SU_2	SU_4
⋮	⋮	⋮	⋮	⋮	⋮	⋮
14	SU_2	SU_5	SU_6	SU_4	SU_1	SU_3

Step 4: Evaluating the different scheduling options

Table VI shows the undiscounted cash flow elements of each MMF and AE in the model introduced in Figure 2 as estimated by L_RU's project team.

Because SU_5 generates revenue according to the number of customers in the L_RU's customer database (see Step 3), its cash flow elements vary according to the context in which SU_5 is developed. The greater the number of customers in the database, the higher the revenue generated by SU_5. Therefore

- If SU_1 precedes the development of SU_5, which is indicated by $SU_1 \rightarrow SU_5$ in Table VI, then SU_5 yields US \$60 thousand in the second period, US \$90 thousand in the third period, US \$130 in the fourth period and so on;
- If it is SU_2 that precedes the development of SU_5, i.e. $SU_2 \rightarrow SU_5$, then SU_5 yields smaller values each period. It yields US \$ 20 thousand in the second period, US \$ 30 thousand in the third period, US \$ 50 in the fourth period and so on;
- However, if both SU_1 and SU_2 precede the development of SU_5, i.e. $(SU_1,SU_2) \rightarrow SU_5$, then the value yielded by SU_5 in each period reaches its highest values. SU_5 yields US \$ 90 thousand in the second period, US \$ 120 thousand in the third period, US \$ 180 in the forth period and so on.

TABLE VI.
SOFTWARE-UNIT UNDISCOUNTED CASH-FLOW ELEMENTS

Cash Flow Elements (US $ 1,000)					
Id	Period				
	1	2	3	4 to 14	15
SU_1	-50	50	70	100	50
SU_2	-70	20	28	40	20
SU_3	-30	500	700	1,500	1,000
SU_4	-40	200	280	600	200
$SU_1 \rightarrow SU_5$	-80	60	90	130	60
$SU_2 \rightarrow SU_5$	-80	20	30	50	20
$(SU_1,SU_2) \rightarrow SU_5$	-80	90	120	180	90
SU_6	-10	0	0	0	0

In all circumstances the investment required by SU_5 remains the same, i.e. US \$ 80 thousand. Table VII shows the net present value (NPV) of each MMF and AE presented in Figure 2 according to the period in which their development starts, considering an interest rate of 2% per period and that, due to shortage of funds, only one development team has been allocated to work in the project.

TABLE VII.
SOFTWARE-UNIT NET PRESENT VALUE

Net Present Value (US $ 1,000)				
Id	Period			
	1	2	$\cdots$	6
SU_1	1,045	987	$\cdots$	623
SU_2	368	346	$\cdots$	204
SU_3	16,001	14,944	$\cdots$	9,478
SU_4	6,221	5,950	$\cdots$	3,766
$SU_1 \rightarrow SU_5$	1,334	1,263	$\cdots$	792
$SU_2 \rightarrow SU_5$	454	430	$\cdots$	253
$(SU_1,SU_2) \rightarrow SU_5$	1,885	1,781	$\cdots$	1,126
SU_6	-10	-10	$\cdots$	-9

Table VIII relates the scheduling options introduced in Table V to their respective NPV. For example, in the seventh option SU_1 is developed first. Next, SU_5, SU_6, SU_3, SU_2 and SU_4 are developed in the second, third, forth, fifth and sixth periods respectively, yielding an NPV of \$18,460 thousand, which the highest NPV among all scheduling options and, as a result, the logical choice for the L_RU's project team.

Obviously this NPV can only be reached because L_RU is a well established company that runs its loan business with the support of trustworthy business processes, which can be temporarily used to support the development of the salary-loan software. For example, when the development

of SU_1 or SU_2 is completed, applying customers do not need to wait for the development of the remaining modules to be informed of the conditions in which they may be granted a loan. At this point L_RU's loan processes take over and provide the adequate answer. This holds true for any combination of previously developed modules.

Step 5: Dealing with the nonessential software units

Common sense and experience has shown that there are two major ways of loosing money in the loan business: lending money to those who will not pay back their loans and failing to lend money to those who will [17]. Therefore, before committing to a particular scheduling option based upon a model that only considers essential software units, the L_RU's project team decided to examine the effect of nonessential software units that deal with these two different aspects of the loan business on the returns yielded by the salary-loan software. Table IX describes the nonessential software units conceived by the project team.

There is only one nonessential minimum-marketable-feature module (NMMF) among the nonessential software units, i.e, SU_7, "Thorough credit analysis", which further enriches the L_RU's customer database with information that opens opportunities for new marketing campaigns from both L_RU's and its associated companies.

As the revenue generated by SU_7 depends upon the number of trustworthy customers it is able to identify, the more customers who have their credit extensively analyzed, the better. Hence the revenue generated by SU_7 is influenced by the development of both SU_1,"Apply for a loan", and SU_2, "Apply for refinancing", in the same way that SU_5, "Quick credit analysis", is. Table X presents the cash flow elements of SU_7. The revenue generated by this particular software unit stems from the premium fee paid by L_RU's associated companies to access the enrichment customer database.

Surprisingly, this is not the only way in which SU_7 contributes to the value of the salary-loan project. If implemented, "Thorough credit analysis" increases the number of approved loans and refinancing applications. The more loans and refinancing applications are approved, the higher the returns yielded by "Accept loan conditions" and "Accept refinancing conditions". Therefore, "Thorough credit analysis" also helps to increase the revenue generated by these two MMFs.

TABLE VIII.
SCHEDULING OPTION NPV

Scheduling Option	NPV (US $ 1,000)
1	17,564
2	16,769
⋮	⋮
7	18,460
⋮	⋮
14	15,814

TABLE IX.
NONESSENTIAL SOFTWARE UNIT DESCRIPTIONS

Software Units		
Id	**Name**	**Description**
SU_7	Thorough credit analysis	Performs a more detailed credit analysis on applications that have been turned down by the "Quick credit analysis" software unit, with the intention of granting a loan to applying customers, if this turns out to be possible
SU_8	Contract life insurance	With the intention of granting loans, include the cost of life insurance in loan applications turned down by the "Quick credit analysis" software unit due to the short life expectance of loan applicants
SU_9	Handle approved loans	Determines the earliest possible date on which L_RU can grant a loan or a refinancing request that has been turned down by the "Check for funding availability" due to temporary lack of funds

TABLE X.
SU_7 CASH-FLOW ELEMENTS

Cash Flow Elements (US $ 1,000)					
Id	**Period**				
	1	**2**	**3**	**4 to 9**	**10**
$SU_1 \rightarrow SU_7$	-90	65	75	90	65
$SU_2 \rightarrow SU_7$	-90	25	30	35	25
$(SU_1, SU_2) \rightarrow SU_7$	-90	90	100	130	90

On the other hand, "Handle approved loans" and "Contract life insurance" are self-contained units whose services neither customers nor associated companies are willing to pay for. Therefore, they are indeed nonessential architectural elements, or NAEs for short. Even though these two software units are unable to generate revenue on their own, they do contribute to the value of the salary-loan software by influencing the revenue generated by other MMFs.

In a similar fashion to "Thorough credit analysis", both "Handle approved loans" and "Contract life insurance" increase the number of approved loans and refinancing applications. As a result, they further improve the revenue generated by both "Accept loan conditions" and "Accept refinancing conditions". Table XI presents the estimated impact of the development of nonessential software units on the revenue generated by essential software units.

TABLE XI.
THE ESTIMATED FINANCIAL IMPACT OF NONESSENTIAL SOFTWARE UNITS.

		Essential Software Units	
		SU_3 (Acpt. loan cond.)	**SU_4** (Acpt. refinancing cond.)
Non-essential Software Units	**SU_7** (Thorough credit analysis)	+20%	+15%
	SU_8 (Contract life insurance)	+15%	+10%
	SU_9 (Handle approved loans)	+25%	+20%

For example, according to the information displayed in Table XI, from the moment "Thorough credit analysis" is implemented the revenue generated by "Accept loan

conditions" and "Accept refinancing conditions" increase 20% and 15% respectively.

Step 6: Replanning the software implementation

Figure 3 introduces the new precedence diagram for the salary-loan software. This diagram takes into consideration the existence of both essential and nonessential software units. A gray background with a dashed-line frame around a software unit identification tag such as SU_7 nor SU_8 indicates a nonessential software unit, which may or may not be developed.

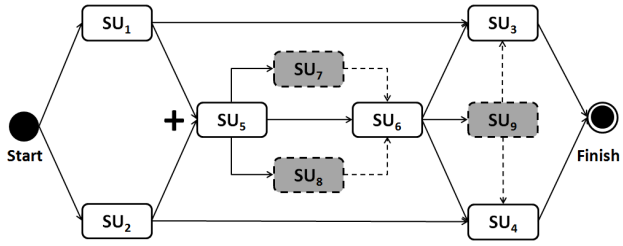


Figure 3. A precedence diagram containing both essential and nonessential software units.

However, if a nonessential software unit is developed, than the dependencies that it creates must be complied with. For example, if neither SU_7 nor SU_8 are developed, than the development of SU_6 can start right after the development of SU_5 is completed. Nevertheless, if SU_7 is developed and SU_8 is not, then the development of SU_6 can only start when the development of both SU_5 and SU_7 are completed, and so on. Table XII presents the scheduling options for the salary-loan software considering the existence of both essential and nonessential software units. There are 144 different scheduling options altogether.

TABLE XII.
NEW SCHEDULING OPTIONS

#	Period								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	SU_1	SU_2	SU_5	SU_6	SU_3	SU_4			
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
81	SU_2	SU_1	SU_5	SU_7	SU_8	SU_6	SU_3	SU_4	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
144	SU_2	SU_5	SU_7	SU_9	SU_6	SU_1	SU_8	SU_4	SU_3

Step 7: Which scheduling option does provide the highest NPV?

Table XIII shows the undiscounted cash flow elements of each MMF, AE, NMMF and NAE introduced in Figure 3, as estimated by L_RU's project team.

Table XIV shows the net present value (NPV) of each MMF, AE, NMMF and NAE presented in Figure 3 according to the period in which their development starts, considering an interest rate of 2% per period. It should be noted that the number of periods that takes to develop the whole software has potentially increased from six to nine periods, because of the possible introduction of nonessential MMFs and AEs (check Table XII).

TABLE XIII.
NEW SOFTWARE-UNIT CASH-FLOW ELEMENTS

New Cash Flow-Elements (US $ 1,000)					
Software Unit Id	**Period**				
	1	**2**	**3**	**4 to 14**	**15**
SU_1	-50	50	70	100	50
SU_2	-70	20	28	40	20
SU_3	-30	500	700	1,500	1,000
$SU_7 \rightarrow SU_3$	-30	600	840	1,800	1,200
$SU_8 \rightarrow SU_3$	-30	625	875	1,875	1,250
$SU_9 \rightarrow SU_3$	-30	575	805	1,725	1,150
$(SU_7,SU_8) \rightarrow SU_3$	-30	725	1,015	2,175	1,450
⋮	⋮	⋮	⋮	⋮	⋮
SU_9	-10	0	0	0	0

TABLE XIV.
ESSENTIAL AND NONESSENTIAL SOFTWARE UNIT NPV

Net Present Value (US $ 1,000)				
Software Unit Id	**Period**			
	1	**2**	⋯	**15**
SU_1	1,045	987	⋯	369
SU_2	368	346	⋯	105
SU_3	16,001	14,944	⋯	5,653
$SU_7 \rightarrow SU_3$	19,207	17,939	⋯	6,788
$SU_8 \rightarrow SU_3$	20,009	18,688	⋯	7,072
$SU_9 \rightarrow SU_3$	18,406	17,190	⋯	6,504
$(SU_7,SU_8) \rightarrow SU_3$	23,215	21,683	⋯	8,208
⋮	⋮	⋮	⋮	⋮
SU_9	-10	-10	⋮	-9

Among all the scheduling options introduced in Table XII, the most logical option for L_RU is to develop the software units comprising the salary-loan software in the following order:

$$SU_2 \rightarrow SU_1 \rightarrow SU_5 \rightarrow SU_7 \rightarrow SU_8 \rightarrow SU_6 \rightarrow SU_3 \rightarrow SU_4,$$

which yields an NPV of US $ 19,651. One should note that this scheduling option does not contemplate the development of SU_9, "Handle approved loan". According to the estimates generated by L_RU's project team, the cost of developing SU_9 is not compensated by its positive impact on the revenue generated by SU_3 and SU_4.

IV. CONCLUSIONS

Table XV compares the value of the different alternatives considered by L_RU's project team for the development of the salary-loan software.

Observe that, by and large, what nonessential self-contained software units and flexible precedence relations do is to allow project managers to postpone the development of less valuable software units until this becomes imperative. As a result, the value of the more profitable developments paths may be pursued and appropriated earlier.

For example, in the salary-loan software project the precedence relations of "Quick credit analysis" is flexible in the sense that the development of this software unit may start when the development of either "Apply for a loan" or "Apply for refinancing" is completed, or even when the development of both are completed. This flexibility

alone increases the value of the salary-loan software from US $17,564 to US $18,460 thousand at no extra cost. Subsequently, with the introduction of more flexibility in the form of the nonessential MMFs and AEs the project value jumps to US $19,651 thousand.

All of this supports the claim that the introduction of flexible precedence relations as well as the presence of NMMFs and NAEs among essential software units is likely to increase the business value of software projects.

Identifying nonessential software units is not as hard as it may seem at first sight. If one has specific goals to be reached by a software project that are clearly stated, non essential software units are fundamentally those that can be taken away from the project without hindering the likelihood of it fulfilling its stated goal [18], [19].

However, despite the considerable body of research that is available in the literature on the impact of flexibility upon the valuation, selection and management of software projects [5], [15], [20], the introduction of flexibility via nonessential self-contained software units and flexible precedence relations has remained largely unexplored.

This paper goes forward in filling this gap by helping organizations to get the most out of their software projects, and, as a result, to increase their level of efficiency in the use of information technology. By being conscious of the potential value brought by nonessential self-contained software units and flexible dependency relations, companies can develop larger and more complex projects from a relatively smaller investment.

REFERENCES

[1] W. R. Priya Kurien and V. S. Purushottam, "The case for re-examining IT effectiveness," *Journal of Business Strategy*, vol. 25, no. 2, pp. 29–36, 2004.

[2] M. A. Cusumano, *The Business of Software*. Free Press, 2004.

[3] M. Denne and J. Cleland-Huang, "Financially informed requirements prioritization," in *Proceedings of the 27^{th} International Conference on Software Engineering (ICSE)*. St. Louis, MO, USA: ACM Press, 15-21 May, 2005, pp. 710–711.

[4] W. V. Grembergen, *Information Technology Evaluation Methods and Management*. Idea Group Publishing, 2001.

[5] M. Benaroch, S. Shah, and M. Jeffery, "On the valuation of multistage information technology investments embedding nested real options," *Journal of Management Information Systems*, vol. 23, no. 1, pp. 239–261, Summer 2006.

TABLE XV.
ALTERNATIVES CONSIDERED BY THE DEVELOPMENT TEAM OF THE SALARY-LOAN SOFTWARE

#	Preced. Diag.	Flexibility	Logical Choice NPV ($1,000)
1	Fig. 1	None	$17,564
2	Fig. 2	The precedence dependency relation between SU_1, SU_2 and SU_5 is acknowledged to be flexible	$18,460
3	Fig. 3	The precedence dependency relation between SU_1, SU_2 and SU_5 is acknowledged to be flexible, and NMMFs and NAEs are introduced	$19,651

[6] S. Dekleva, "Justifying investments in IT," *Journal of Information Technology Management*, vol. XVI, no. 3, pp. 1–8, 2005.

[7] R. J. Benson, T. Bugnitz, and B. Walton, *From Business Strategy to IT Action: Right Decisions for a Better Bottom Line*. Wiley, 2004.

[8] K. Milis and R. Mercken, "The use of the balanced scorecard for the evaluation of information and communication technology projects," *International Journal of Project Management*, vol. 22, no. 2, pp. 87–97, February 2004.

[9] K. M. Rosacker and D. L. Olson, "An empirical assessment of IT project selection and evaluation methods in state government," *PM Journal*, vol. 29, no. 1, pp. 49–58, February 2008.

[10] A. J. Alencar, E. A. Schmitz, and E. P. de Abreu, "Maximizing the business value of software projects," in *10^{th} Inter. Conf. on Enterprise Info. Systems.*, vol. ISAS-2. Barcelona, Spain: INSTICC, June 12-16, 2008, pp. 162–169.

[11] M. Denne and J. Cleland-Huang, "The incremental funding method: Data-driven software development," *Software, IEEE*, vol. 21, no. 3, pp. 39– 47, 2004.

[12] D. W. Hubbard, *How to Measure Anything*. Wiley, 2007.

[13] A. A. Groppelli and E. Nikbakht, *Finance*, 5^{th} ed. Barron's, 2006.

[14] C. L. Peterson, "Usury law, payday loans, and statutory sleight of hand," *Minnesota Law Review*, vol. 92, no. 4, April 2008.

[15] R. G. Fichman, M. Keil, and A. Tiwana, "Beyond valuation: Real options thinking in IT project management," *California Management Review*, vol. 47, no. 2, pp. 74–100, 2005.

[16] C. Hibbs, S. Jewett, and M. Sullivan, *The Art of Lean Software Development: A Practical and Incremental Approach*. O'Reilly Media, 2009.

[17] N. Siddiqi, *Credit Risk Scorecards*. Wiley, October 2005.

[18] K. Pohl, G. Bckle, and F. J. van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. NY, USA: Springer, November 2010.

[19] P. Valente, *Goals Software Construction Process*. VDM, 2009.

[20] L.-C. Wu and C.-S. Ong, "Management of information technology investment: A framework based on a real options and mean–variance theory perspective," *Technovation*, vol. 28, no. 3, pp. 122–134, 2008.

Antonio J. Alencar is a researcher with the Federal University of Rio de Janeiro (UFRJ), Brazil. He received his B.Sc. in Mathematics and M.Sc. in System Engineering and Computer Science from UFRJ. He holds a D.Phil. in Computer Science from Oxford University, England. His research interests include economics of software engineering, IT strategy and risk analysis.

Rafael A. Nascimento is an M.Sc. student with the Federal University of Rio de Janeiro, Brazil. His research interests include software development methodologies and economics of software engineering.

Eber A. Schmitz is a Professor of Computer Science with the Federal University of Rio de Janeiro. He holds a B.Sc. in Electrical Engineering from the Federal University of Rio Grande do Sul, an M.Sc. in Electrical Engineering from the Federal University of Rio de Janeiro and a Ph.D. in Computer Science and Control from the Imperial College of Science, Technology and Medicice, England. His research interests include software modeling tools, business process modeling and stochastic modeling.

Alexandre L. Correa is a Professor of Computer Science with the Federal University of the State of Rio de Janeiro. He holds a B.Sc. in Mathematics and an M.Sc. and a Ph.D. in System Engineering and Computer Science from the Federal University of Rio de Janeiro. His research interests include reverse engineering, system validation and software development tools.

Angélica F. S. Dias is Ph.D. candidate with the Federal University of Rio de Janeiro (UFRJ). She holds a M.Sc. in Computer Science from UFRJ. Her research interests include human-computer interaction and the social impact of new technologies.

Investigation of Aspect-Oriented Metrics for Stability Assessment: A Case Study

Mahmoud O. Elish, Mojeeb Al-Khiaty, Mohammad Alshayeb
Information & Computer Science Department
King Fahd University of Petroleum & Minerals
Dhahran, Saudi Arabia
Emails: {elish, alkhiaty, alshayeb@kfupm.edu.sa

***Abstract* — Stability assessment provides software managers with early insight into trends in software evolution, and thus assists them in managing and controlling long-lived software systems. However, there are few empirical studies that have been conducted to relate software metrics with external quality attributes of aspect-oriented software in general, and metrics have not been evaluated as indicators of aspect stability in particular. This paper investigates the relationships between 13 aspect-oriented metrics and aspect stability. These metrics measure different structural properties of an aspect: size, coupling, cohesion, and inheritance. A case study was conducted using an open source aspect-oriented software consisting of 76 aspects. The results obtained from this study indicate statistically significant correlation between most of the size metrics and aspect stability. The cohesion metric was also found to be significantly correlated with aspect stability. In addition, different prediction models were built using different combinations of metrics' categories. It was observed that the best accuracy was achieved as a function of some of the size and inheritance metrics.**

***Index Terms* — software metrics, software stability, aspect-oriented software.**

I. INTRODUCTION

Separation of concerns is one of the vital principles in software engineering for achieving quality software [4]. Although different approaches — including object-oriented programming, component-oriented programming, and design patterns — provide useful modularity mechanisms, none of them satisfactorily modularize all concerns of complex software systems. Some of the concerns still inherently crosscut the modularity of multiple modules and are difficult to be captured by these techniques [6]. Aspect-Oriented Software Development (AOSD) provides an explicit concepts and mechanisms for separating the crosscutting concerns [6]. It is increasingly getting popularity as useful practice to improve the modularization of software artifacts.

It is vital to quantitatively assess the quality of software produced using AOSD. In this regard, software metrics are needed to do such assessment. The external quality attributes of aspect-oriented software are usually assessed, using modeling techniques, as a function of metrics that measure the internal structural properties of the aspect-oriented software. However, there are few empirical studies [1, 7, 8, 10] that have been conducted to relate software metrics with external quality attributes of aspect-oriented software in general, and metrics have not been evaluated as indicators of aspect stability in particular.

Motivated by the foregoing issues, in this paper, we empirically investigate the relationships between a suite of aspect-oriented metrics and aspect stability. In other words, this paper investigates whether or not the metrics under investigation are good early indicators of aspect stability in aspect-oriented software. Aspect stability, in this paper, refers to the extent to which the revisions made to an aspect are infrequent. The availability of adequate metrics for stability assessment provides software managers early insight into trends in software evolution, and thus assists them in managing and controlling long-lived software systems.

The rest of this paper is organized as follows Section II defines the aspect-oriented metrics under investigation. Section III discusses the case study and its results. Section IV reviews related work. Section V concludes the paper.

II. ASPECT-ORIENTED METRICS

In this study, 13 aspect-level metrics are investigated as indicators of aspects' stability. These metrics were chosen because: (i) they measure different structural properties of an aspect: size, coupling, cohesion, and inheritance; and (ii) they refine classical object-oriented metrics, such as C&K metrics [3], which are well-established and based on sound measurement theory. The metrics are defined next.

A. Size Metrics

- Number of Attributes (NA) [7]: The NA metric of an aspect is defined as the number of attributes defined in the aspect.
- Number of Methods (NM): The NM metric of an aspect is defined as the number of methods defined in the aspect.

doi:10.4304/jsw.6.12.2508-2514

- Number of Advices (NAD) [9]: The NAD metric of an aspect is defined as the number of advices in the aspect.
- Number of Pointcuts (NP) [9]: The NP metric of an aspect is defined as the number of pointcuts in the aspect.
- Number of Introductions (NI) [9]: The NI metric of an aspect is defined as the number of introductions (intertype declarations) in the aspect.
- Lines of Code (LOC) [7]: The LOC metric of an aspect is the number of lines of code, excluding comment and blank lines, in the aspect.

B. *Coupling Metrics*

- Afferent Coupling through Introductions (ACTI): The ACTI metric of an aspect is defined as the number of classes and aspects that are affected by the aspect through introductions.
- Afferent Coupling through Pointcuts (ACTP): The ACTP metric of an aspect is defined as the number of classes and aspects that are affected by the aspect through pointcuts.
- Efferent Coupling through Introductions (ECTI): The ECTI metric of an aspect is defined as the number of aspects that affect the aspect through introductions.
- Efferent Coupling through Pointcuts (ECTP): The ECTP metric of an aspect is defined as the number of aspects that affect the aspect through pointcuts.

C. *Cohesion Metric*

- Lack of Cohesion in Operations (LCO) [7]: The LCO metric of an aspect is defined as the number of method/advice pairs that do not access the same instance variable.

D. *Inheritance Metrics*

- Depth of Inheritance (DIT) [7]: The DIT metric of an aspect is defined as the level of the aspect in its inheritance hierarchy, i.e. the length of the longest path from the aspect to the root of inheritance tree.
- Number of Children (NOC) [2]: The NOC metric of an aspect is defined as the number of immediate sub-aspects of the aspect, i.e. the number of aspects that inherit directly from the aspect.

III. THE CASE STUDY

The main objective of this case study is to investigate the relationships between the 13 aspect-level metrics, described in the previous section, and aspect stability. For this purpose, a large open source aspect-oriented software system called Glassbox[1] was chosen because of the availability of its revision history. The system, which is written in Java and ApectJ, is an automated troubleshooting and monitoring agent for Java applications. It consists of 76 aspects and about 420 classes.

A. *Descriptive Statistics*

The 13 aspect-oriented metrics were collected from each aspect in the first version of the Glassbox system. Table 1 provides their descriptive statistics. It can be observed that aspects vary in size in terms of the number of attributes, methods, advices, pointcuts, introductions, etc. Furthermore, there is a relatively good utilization of inheritance, and aspects' afferent couplings are more than their efferent couplings.

The dependent variable is aspect stability. The number of revisions made to an aspect was used as a proxy measure for its stability, i.e. the more revisions the less stable it is. It was collected using the Concurrent Versions System (CVS) repository of the Glassbox system. The log data (i.e. revision history) for each aspect was obtained by using the 'log' subcommand of 'cvs' command, which contains the total number of revisions made to it. These revisions might be corrective, perfective, adaptive, and/or preventive. In Glassbox system, the number of revisions made to each aspect varies from 1 to 9 with an average of 1.7.

TABLE 1.
DESCRIPTIVE STATISTICS

Metric	Min	Max	Avg.	Std. Dev.
NA	0	9	1.29	1.91
NM	0	31	3.03	5.62
NAD	0	12	1.87	2.45
NP	0	29	2.91	4.01
NI	0	13	0.99	2.49
LOC	2	283	44.95	57.95
ACTI	0	381	7.09	45.61
ACTP	0	500	12.08	59.72
ECTI	0	28	1.39	3.11
ECTP	0	3	2.28	0.74
LCO	0	219	4.93	28.36
DIT	0	5	1.45	1.72
NOC	0	10	0.36	1.45

B. *Correlation Analysis*

The correlation analysis aims to determine if each individual aspect-oriented metric is significantly related to aspect stability. For this purpose, Spearman's rank correlation was performed due to the nonparametric nature of the metrics. The significance of the correlation was tested at 99% confidence level (i.e. p-level ≤ 0.01). The results obtained by applying this analysis are given in Table 2, where bold values indicate statistically significant correlations. All size metrics, except the NM metric, were found to be significantly correlated with aspect stability. In addition, the cohesion metric (LCO) was also found to be significantly correlated with aspect stability. This is not the case, however, for the coupling and inheritance metrics.

[1] http://sourceforge.net/projects/glassbox/

TABLE 2.
SPEARMAN CORRELATION ANALYSIS RESULTS

Metric	Correlation Coefficient (r)	p-value
NA	**0.375**	**<0.01**
NM	0.177	0.125
NAD	**0.474**	**<0.01**
NP	**0.479**	**<0.01**
NI	**0.361**	**<0.01**
LOC	**0.629**	**<0.01**
ACTI	0.159	0.169
ACTP	0.167	0.150
ECTI	-0.099	0.394
ECTP	0.112	0.335
LCO	**0.305**	**<0.01**
DIT	0.130	0.264
NOC	0.034	0.768

C. *Multivariate Regression Analysis*

Multivariate Linear Regression (MLR) is the most commonly used technique for modeling the relationship between two or more independent variables and a dependent variable by fitting a linear equation to observed data. The main advantages of this technique are its simplicity and that it is supported by many popular statistical packages. The multivariate analysis was performed to construct different MLR prediction models for predicting aspect stability as a function of the 13 metrics (independent variables) under investigation. Since there are four categories of these metrics (size, coupling, cohesion, and inheritance), 15 different prediction models were built that represent all possible combinations of these categories (see Table 3).

TABLE 3.
CATEGORY(S) OF METRICS IN EACH MODEL

Model	Size Metrics	Coupling Metrics	Cohesion Metric	Inheritance Metrics
M1	√			
M2		√		
M3			√	
M4				√
M5	√	√		
M6	√		√	
M7	√			√
M8		√	√	
M9		√		√
M10			√	√
M11	√	√	√	
M12	√	√		√
M13	√		√	√
M14		√	√	√
M15	√	√	√	√

1) *Variables Selection*

Variables selection is a preliminary step used in multivariate data analysis. When there are many independent variables, there is a possibility that some of these variables contain redundant or noisy information. Additionally, there can be a high correlation between independent variables which can adversely affect the results meanwhile it does not add new information. Thus, it is useful to reduce the number of independent variables and remove the collinearity in each model. In this study, we performed the best first search method in WEKA machine learning toolkit for variables selection [11]. It searches the space of variable subsets by greedy hill climbing augmented with a backtracking facility.

The selected independent variables and their regression coefficient for all the 15 prediction models are provided in Table 4. An empty cell indicates that the corresponding metric is not one of the independent variables in the corresponding model. It can be observed that some models are the same (M1 and M6; M5 and M11; M7 and M13; M12 and M15) due to the variables selection step. This means that there are 11 different models out of the 15 models, which will be considered in the subsequent analyses.

There are some interesting observations that can be obtained from Table 4. First, only NP and LOC metrics were selected from the size metrics category. Moreover, the impact of the NP metric is stronger than the LOC metric because the regression coefficient of the NP metric is higher than the regression coefficient of the LOC metric. Second, the only coupling metric that was selected is the ACTI metric, but its regression coefficient is weak. Third, the LCO metric was selected with coupling and inheritance metrics, but not with size metrics. Finally, the NOC metric had the strongest impact (i.e. highest regression coefficient) on the prediction models.

2) *Models' Goodness of Fit*

In order to measure and evaluate the goodness of fit for each model, we used R^2 (R-squared). It indicates what percentage of the variability in the dependent variable can be explained by the independent variables in each model. The R^2 value for each prediction model is provided in Table 5 and visualized by Figure 1. The M12 model (based on size, coupling and inheritance metrics) has the highest R^2 value (0.625), whereas the M4 model (based on the inheritance metric only) has the lowest R^2 value (0.042). Another interesting observation is that those models that include size metrics have better R^2 values compared to those that do not include them. It is also interesting to observe that improved R^2 value could be achieved by considering other structural properties in addition to size. For instance, improved R^2 value was achieved by M5, M7 and M12 models compared to the M1 model.

3) *Cross Validation*

A leave-one-out cross-validation procedure was used to evaluate and compare the accuracy of the prediction models. In this procedure, one observation is removed from a dataset with *n* observations, and then each prediction model is built with the remaining *n-1* observations and evaluated in predicting the value of the observation that was removed. The process is repeated *n* times; each time removing a different observation.

TABLE 4.
SELECTED METRICS AND THEIR REGRESSION COEFFICIENT FOR THE REGRESSION MODELS

Model		Size Metrics						Coupling Metrics				Cohesion Metric	Inheritance Metrics	
	Constant	NA	NM	NAD	NP	NI	LOC	ACTI	ACTP	ECTI	ECTP	LCO	DIT	NOC
M1	1.154				0.158		0.014							
M2	2.154							0.010						
M3	2.074											0.030		
M4	2.138													0.241
M5	1.155				0.147		0.014	0.005						
M6 (same as M1)	1.154				0.158		0.014							
M7	1.072				0.148		0.014							0.237
M8	1.998							0.010				0.031		
M9	1.791							0.011					0.191	0.221
M10	1.976											0.031		0.267
M11 (same as M5)	1.155				0.147		0.014	0.005						
M12	1.070				0.135		0.014	0.005						0.244
M13 (same as M7)	1.072				0.148		0.014							0.237
M14	1.673							0.011				0.031	0.158	0.253
M15 (same as M12)	1.070				0.135		0.014	0.005						0.244

TABLE 5.
MODELS' GOODNESS OF FIT

Model	R^2
M1 (also M6)	0.568
M2	0.069
M3	0.255
M4	0.042
M5 (also M11)	0.583
M7 (also M13)	0.608
M8	0.331
M9	0.150
M10	0.306
M12 (also M15)	0.625
M14	0.410

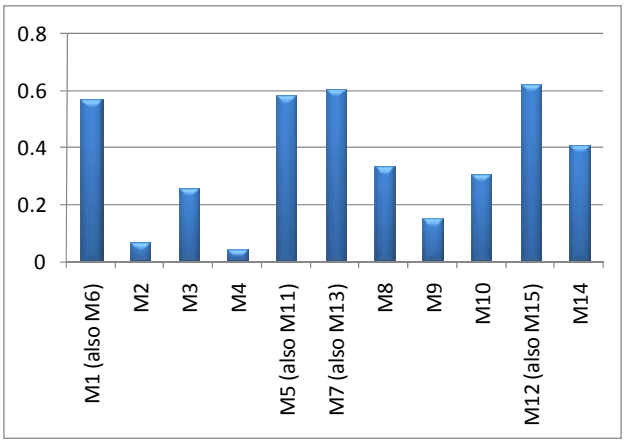


Figure 1. Models' R-squared.

The accuracy of the prediction models were evaluated based on de facto standard and commonly used measures: mean magnitude of relative error (MMRE) and prediction at level q (Pred(q)) measures. MMRE over a dataset of n observations is calculated as follows:

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i$$

where MRE_i is a normalized measure of the discrepancy between the actual value (x_i) and the predicted value ($\hat{x}_i$) of observation i. It is calculated as follows:

$$MRE_i = \frac{|x_i - \hat{x}_i|}{x_i}$$

Pred(q) is a measure of the percentage of observations whose MRE is less than or equal to q. It is calculated as follows:

$$Pred(q) = \frac{k}{n}$$

where k is the number of observations whose MRE is less than or equal to a specified level q, and n is the total number of observations in the dataset. In this study, we used Pred(0.25), which is commonly used in the literature.

Table 6 shows the MMRE and Pred(0.25) values for the prediction models, and Figure 2 and Figure 3 visualize them respectively. It can be noticed that the M1 model (based on size metrics only) achieved the best MMRE value (0.483), followed by the M7 model (based on size and inheritance metrics) which achieved very competitive MMRE value (0.489). The best model in terms of Pred(0.25) is M7, which achieved a value of 34.2%. Based on these results and from Table 4, we can conclude that the best prediction of aspect stability can be achieved as a function of the NP, LOC and NOC metrics. This means that the higher the number of pointcuts, lines of code, and number of children of an aspect, the less stable it will be (i.e. undergo more revisions). The inclusion of the coupling and cohesion metrics and other

size metrics under investigation is not useful for aspect stability prediction.

TABLE 6.
EVALUATION OF THE PREDICTION MODELS

Model	MMRE	Pred(0.25)
M1 (also M6)	0.483	30.3%
M2	0.731	22.4%
M3	0.673	22.4%
M4	0.840	21.1%
M5 (also M11)	0.514	26.3%
M7 (also M13)	0.489	34.2%
M8	0.637	23.7%
M9	0.739	26.3%
M10	0.699	22.4%
M12 (also M15)	0.619	30.3%
M14	0.835	26.3%

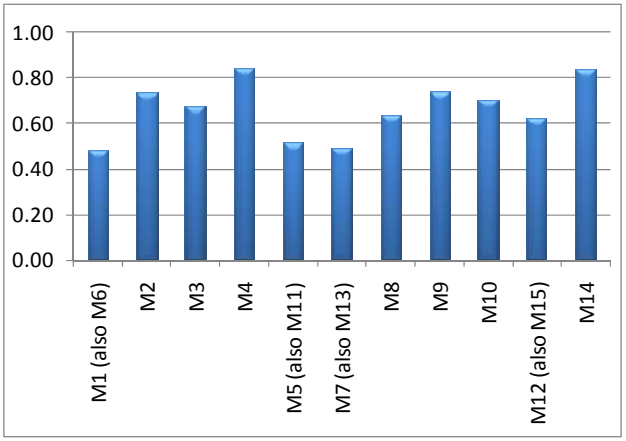


Figure 2. Models' MMRE.

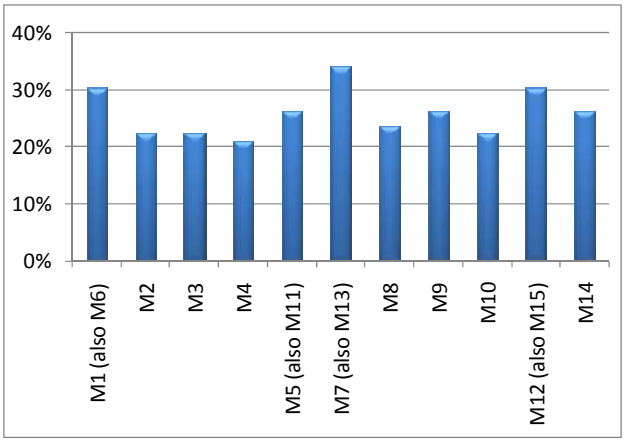


Figure 3. Models' Pred(0.25).

D. *Threats to Validity*

This case study evaluated one open source software. The SourceForge[2] open source software repository was searched for suitable aspect-oriented software but it was challenging because: (1) the number of aspect-oriented software was limited since it is a relatively new paradigm; (2) most of the existing software have either limited number of aspects or no detailed revision history. More future case studies should be conducted, as more open source software with detailed revision history become available, to further support the findings of this paper and to accumulate knowledge.

In this study, aspect stability was measured as the total number of revisions made to it, i.e. the frequency of revisions. Other dimensions of stability can be addressed as future work. In addition, this study was a regression and correlation study. Association between some of the investigated metrics and aspect stability was observed but causality of the association cannot be claimed.

IV. RELATED WORK

There are few research studies that have explored the relationships between aspect-oriented metrics and external software quality attributes [1, 7, 8, 10]. Table 7 summarizes those studies by listing the relevant metrics and the external software quality attributes against which they were empirically evaluated. Cells marked with (√) indicate that significant correlations were observed, whereas cells marked with (X) indicate no significant correlations. Unmarked cell indicates that the corresponding metric was not empirically evaluated against the corresponding external software quality attribute.

Greenwood et al. [5] investigated the impact of aspectual decompositions on design stability. They evaluated the design stability of two different implementations (aspect-oriented and object-oriented) of a real-life web-based information system by applying different types of maintenance tasks. The stability was assessed using traditional suites of modularity and change impact metrics. That is, the more stable design is the one that minimizes the undesirable variation in the values of the metrics after applying the maintenance tasks. The overall conclusion of that study is that "aspect decomposition narrows the boundaries of concern dependencies, however, with more tight and intricate interactions" [5].

The objective of this paper is different from the work of Greenwood et al. [5]. We have investigated a set of metrics as early indicators (predictors) of aspect stability, whereas they compared the design stability of aspect-oriented implementation vs. object-oriented implementation.

[2] http://sourceforge.net/

TABLE 7.
RELATIONSHIPS BETWEEN ASPECT-ORIENTED METRICS AND EXTERNAL SOFTWARE QUALITY ATTRIBUTES

Metrics	External Software Quality Attributes						
	Maintainability			Understandability	Reusability	Fault proneness	Stability
	[10]	[7]	[8]	[10]	[7]	[1]	This Paper
Lines of Code	X			X			√
Weighted Operations in Module	X			X		X	
Operation Cohesion	√			√			
Attribute Cohesion	X			X			
Interface Coupling	√			√			
Coupling Between Components		√			√	√	
Depth of Inheritance Tree		√			√	X	X
Concern Diffusion over Components		√			√		
Concern Diffusion over Operations		√			√		
Concern Diffusions over LOC		√			√		
Number of Attributes		√			√		√
Vocabulary Size		√			√		
Coupling on Method Call			X			√	
Coupling on Field Access			X			√	
Response For a Module			X			X	
Coupling on Advice Execution			X			X	
Crosscutting Degree of Aspects			X			√	
Lack of Cohesion in Operations						X	√
Number of Children						X	X
Base-Aspect Coupling						√	
Number of Methods							X
Number of Advices							√
Number of Pointcuts							√
Number if Introductions							√
Afferent Coupling through Introductions							X
Afferent Coupling through Pointcuts							X
Efferent Coupling through Introductions							X
Efferent Coupling through Pointcuts							X

V. CONCLUSION

This paper has investigated the relationships between 13 aspect-oriented metrics and aspect stability. These metrics measure different structural properties of an aspect: size, coupling, cohesion, and inheritance. A case study was conducted using an open source aspect-oriented software consisting of 76 aspects. The results obtained from this study indicate statistically significant correlation between most of the size metrics and aspect stability. The cohesion metric was also found to be significantly correlated with aspect stability. In addition, different prediction models were built using different combinations of metrics' categories. It was observed that the best accuracy was achieved as a function of some of the size and inheritance metrics: number of pointcuts, lines of code, and number of children. More future case studies should be conducted, as more open source software with detailed revision history become available, to further support the findings of this paper and to accumulate knowledge.

This case study contributes interesting preliminary and novel empirical knowledge about the relationships between some aspect-oriented metrics and aspect stability. Future works include exploring more metrics; conducting more case studies; investigating the impact of author styles and other factors on aspect stability; exploring the relationships between aspect-oriented metrics and other software quality attributes; and building computational intelligence models to improve the prediction accuracy.

ACKNOWLEDGMENT

The authors acknowledge the support of King Fahd University of Petroleum and Minerals (KFUPM). This research work was funded by KFUPM under Research Grant No. IN080420.

REFERENCES

[1] R. Burrows, F. Ferrari, A. Garcia, and F. Taïani, "An Empirical Evaluation of Coupling Metrics on Aspect-Oriented Programs," in *ICSE Workshop on Emerging Trends in Software Metrics*, pp. 53-58, 2010.

[2] N. Cacho, C. Sant'Anna, E. Figueiredo, A. Garcia, T. Batista, and C. Lucena, "Composing Design Patterns: A Scalability Study of Aspect-Oriented Programming," in *5th International Conference on Aspect-Oriented Software Development*, pp. 109-121, 2006.

[3] S. Chidamber and C. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Transactions on Software Engineering*, vol. 20, pp. 476-493, 1994.

[4] R. Filman, T. Elrad, S. Clarke, and M. Aksit, *Aspect-Oriented Software Development*: Addison-Wesley, 2005.

[5] P. Greenwood, T. Bartolomei, E. Figueiredo, M. Dosea, A. Garcia, N. Cacho, C. Sant'Anna, S. Soares, P. Borba, U. Kulesza, and A. Rashid, "On the Impact of Aspectual Decompositions on Design Stability: An Empirical Study," in *21st European Conference on Object-Oriented Programming (ECOOP 07), LNCS 4609, Springer*, pp. 176-200, 2007.

[6] R. Laddad, *AspectJ in Action: Practical Aspect-Oriented Programming*: Manning Publications, 2003.

[7] C. Sant'Anna, A. Garcia, C. Chavez, C. Lucena, and A. Staa, "On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework," in *Brazilian Symposium on Software Engineering*, 2003.

[8] H. Shen, S. Zhang, and J. Zhao, "An Empirical Study of Maintainability in Aspect-Oriented System Evolution Using Coupling Metrics," in *2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering*, pp. 233-236, 2008.

[9] K. Srivisut and P. Muenchaisri, "Bad-Smell Metrics for Aspect-Oriented Software," in *6th IEEE/ACIS International Conference on Computer and Information Science*, pp. 1060-1065, 2007.

[10] P. Tonella and M. Ceccato, "Refactoring the aspectizable interfaces: an empirical assessment," *IEEE Transactions on Software Engineering,* vol. 31, pp. 819-832, 2005.

[11] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco: Morgan Kaufmann, 2005.

Mahmoud O. Elish is an assistant professor of Software Engineering in the Information and Computer Science Department at King Fahd University of Petroleum and Minerals, Saudi Arabia. He received the PhD degree in Computer Science from George Mason University, USA. His research interests include software metrics and measurement, software design, software maintenance, empirical software engineering, and software quality predictive models.

Mojeeb Al-Khiaty is a PhD student in computer science and engineering at King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia. He received his Master of Science (MS) degree in Computer Science from KFUPM in June 2009. He obtained his Bachelor of Science (B.S.) degree with honors in Mathematics/Computer from Sana'a University, Yemen in June 1999.

Mohammad Alshayeb is an assistant professor in Software Engineering at the Information and Computer Science Department, King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia. He received his MS and PhD in Computer Science and certificate of Software Engineering from the University of Alabama in Huntsville. He worked as a senior researcher and Software Engineer and managed software projects in the United States and Middle East. He is a certified project manager (PMP). His research interests include Software quality, software measurement and metrics, and empirical studies in software engineering.

Mining a Small Medical Data Set by Integrating the Decision Tree and *t*-test

Ming-Yang Chang
Department of Obstetrics and Gynecology, Chang Gung Memorial Hospital, Taipei, Taiwan 25137, R.O.C.
Email: mychang1126@yahoo.com.tw

Chien-Chou Shih[2],Ding-An Chiang[1] and Chun-Chi Chen[1 *]
[1] Department of Computer Science & Information Engineering, Tamkang University, Tamsui, Taipei County, Taiwan 25137, R.O.C.
[2] Department of Information & Communication, Tamkang University, Tamsui, Taipei County, Taiwan 25137, R.O.C.
Email: ccs@mail.tku.edu.tw, chiang@cs.tku.edu.tw, maurice.chen.tw@gmail.com

***Abstract*—Although several researchers have used statistical methods to prove that aspiration followed by the injection of 95% ethanol left in situ (retention) is an effective treatment for ovarian endometriomas, very few discuss the different conditions that could generate different recovery rates for the patients. Therefore, this study adopts the statistical method and decision tree techniques together to analyze the postoperative status of ovarian endometriosis patients under different conditions. Since our collected data set is small, containing only 212 records, we use all of these data as the training data. Therefore, instead of using a resultant tree to generate rules directly, we use the value of each node as a cut point to generate all possible rules from the tree first. Then, using *t*-test, we verify the rules to discover some useful description rules after all possible rules from the tree have been generated. Experimental results show that our approach can find some new interesting knowledge about recurrent ovarian endometriomas under different conditions.**

***Index Terms*—Data mining, Decision tree, *t*-test, p-value, Ovarian endometriomas**

I. INTRODUCTION

The use of classification algorithms in the medical domains has increasingly been the object of study in recent years [1-3]. Although many conventional treatments are available in clinical medicine for patients suffering from endometriosis, it is a common disease among women of reproductive age with a high recurrence rate, regardless of the treatment type [4]. In recent years, ultrasound-guided aspiration combined with drug therapy has become a new alternative for patients since the ultrasound-guided aspiration of ovarian endometriomas was proposed in 1991 [5]. Therefore, to reduce the high recurrence rate, some medical treatments have combined ultrasound-guided aspiration with tetracycline [6], methotrexate [7], recombinant interleukin-2 [8], or ethanol [9-11].

In recent years, several researchers and our team have used statistical methods to prove that aspiration followed by the injection of 95% ethanol left in situ (retention) is an effective treatment for ovarian endometriomas [11]. They always divide all patients into two groups group 1 (ethanol irrigation) and group 2 (ethanol retention) regardless of other conditions, such as the size of the cyst. In endometriomas-like dataset, the analysis of the influence of treatment effectiveness use statistical and *t*-test cannot integration data mining in related research [12-16] such as *t*-test [12], Logistic regression [15, 17], Decision trees [18-20], and SVM [15, 17]. However, different conditions could generate different recovery rates for the patients, and very few researchers discuss this situation. Therefore, our aim is to investigate recurrent ovarian endometriomas under different conditions. We adopt the statistical method and decision tree techniques together to analyze the postoperative status of ovarian endometriosis patients.

The use of machine learning algorithms for the building of predictive and descriptive data mining models has become widely accepted in medical applications. Various models including Decision trees, Decision rules, Logistic regression, Artificial neural networks, and SVMs have been tested in a wide variety of clinical and medical applications [21]. In order to resolve the endometriomas problem, we should identify and treat the cause of the problem correctly. But such correct diagnosis and treatment require the patients to have extensive live, womb, uterocervical canal, laparoscopy, and others [22]. Computationally, there have been attempts from the endometriomas medical domain to analyze various treatment factors to predict the success of therapy. Previous studies that have utilized *t*-test for endometrial related research have only used this technology to determine the effectiveness of treatment [23]. Based on the decision-tree analysis, the optimal rule to detect the ultrasound characteristics of endometriomas in pre- and postmenopausal patients and to develop rules that characterize endometriomas [20]. For ovarian tumors and pregnancies of unknown location on medical

* Corresponding author.

doi:10.4304/jsw.6.12.2515-2520

decision making (prediction), mathematical algorithms are applied to data sets in order to obtain a model [15, 16]. Kinkel [24] et al. demonstrated in a meta-analysis of indeterminate masses in sonography the superiority of MRT over CT and Doppler sonography in predicting malignancy. Fewer models have been hitherto developed, even though the reliable identification of borderline and ovarian endometriomas would be a good step forward for clinical practice.

The decision tree in classification algorithms has been applied to categorical attributes and numeric attributes in different domains [25]. Since medical data always contain numeric attributes and handling them is a critical task in inductive learning, this task has already been embedded within the decision tree algorithm. Therefore, we adopt the decision tree technique to analyze postoperative status of ovarian endometriosis patients in this study. The decision tree is built by performing a heuristic-based local search to select the best test attribute as the root of the decision tree. So, decision tree creates a branch for each value of that appearing in the training data. Then, the same procedure is operated on each branch to induce the remaining levels of the decision tree until all examples in a leaf belong to the same class. Whereas the tree is represented by a set of rules, each branch of the decision tree is only represented by a rule. However, since each branch of the decision tree is only represented by a rule and the resultant rules of the tree are local, some useful description rules cannot be found by the tree-generation algorithm. Therefore, instead of using a resultant tree to generate classification rules directly, we use the value of each node as a cut point to generate all possible rules from the tree. We can filter out useless rules by using the method of setting minimum recovery rates. However, in the study case, our collected data set is small, containing only 212 records, and we use all of these data as the training data. So because there are different conditions in a possible rule that could generate different recovery rates for the patients and very few researchers discuss this situation, our aim is to investigate the recurrent ovarian endometriomas under the different conditions in possible rule. Under this mining goal, we use *t*-test to verify rules to discover some useful description rules after all possible rules from the tree are generated.

The models were built in collaboration with gynecologists, and resulted in accurate predictions. The ovarian endometriomas models were based on multi-center data and have successfully passed prospective internal and external evaluations. The models for pregnancy of unknown location were based on single center data and have passed the first internal evaluation. However, a large multi-center study is ongoing, aiming for a thorough validation and for the development of new models for pregnancies of unknown location.

II. Materials and Background Knowledge

A. Patients

This study was approved by the Institutional Review Board of our hospital. It was a retrospective review of 212 consecutive patients treated at the outpatient gynecological department of Chang Gung Memorial Hospital, Taipei, Taiwan from July 1994 to July 2008. All patients had undergone previous surgical treatment for ovarian endometriomas and were being seen because of a recurrence. Recurrence was defined as when one or more persistent pelvic cysts greater than 3.0 cm were detected in two consecutive ultrasonographic examinations. Transvaginal ultrasound-guided cyst aspiration and ethanol injection were done on an outpatient basis. Patients randomly received ethanol instillation at 0 min (ethanol injected, then immediately removed), 3–9 min, 10 min, or retention. The procedure was in accordance with that reported previously [8, 23] .

The medical records include basic information of patients, treatment-related information, and clinical examination data obtained from the first visit and three-month, six-month to one year follows-up. The medical data set has 57 attributes, such as age, number of previous pregnancies, degree of menstrual cycle pains, urge technology to help the patient get pregnant, size of the cyst, CA125 blood test value, types of surgery, ethanol irrigation duration. Usually this raw data contained lots of null values (missing information), which might have affected the accuracy of the study. In clinical practice data collection is difficult because the value of attention to patient privacy; so data and collection methods are limited. We required a preprocessing procedure to prepare the valid data for the analysis.

B. Decision Trees

A decision tree is built by selecting the best test attribute as the root of the decision tree. Then, the same procedure is operated on each branch to induce the remaining levels of the decision tree until all examples in a leaf belong to the same class. Decision trees can be categorized by data processing functions into classification trees and regression trees. The common decision tree algorithms are compared in Table I. A classification tree is applied to discrete variables, while a regression tree is applied to quantitative variables. The regression tree was first brought up by Breiman [26] in the introduction of Classification and Regression Trees

TABLE I.
COMPARISON OF DECISION TREE ALGORITHMS

Name	Data attribute	tree type	Partitioning rule	Pruning rule
ID3	Discrete	Classification tree	Information Gain	No Pruning
C4.5	Numeric	Regression tree	Information Gain	Predicted error rate
CHAID	Class	Classification tree	Chi-Square test	No Pruning
CART	Discrete and Numeric	Classification and regression tree	Gini index	Entire error rate

(CART). Usually, in CART analysis, data is categorized into quantitative and discrete data. Quantitative data can be applied to prediction, while discrete data can be applied to classification. In other words, CART analysis is able to simultaneously process both quantitative and discrete data. In the article, we use CART trees in the software of DB2 Intelligent Miner for Data, version 8.1 (IBM Corp., New York), to analyze the collected data to generate the resultant tree.

Ⅲ. MINING USING DECISION TREE AND RESULTS

In the article, we use the CART function in IBM DB2 Intelligent Miner for Data, version 8.1, to analyze the collected data to generate the resultant tree. The first step in reducing the dimensionality is to use PCA algorithm by using the dataset, and decision tree is used for the second feature and rule extraction. After that in each iteration of the loop (step 1) confirmation attributes is selected into the output set (step 3). The selected classification attributes included Cyst_size, CA_125, and BMI. Although the data set has many attributes, as shown in figure 1, only four attributes are selected by the system to build the decision tree. These four attributes are Cyst_size, CA_125, BMI, and Recovery (Recovery means no operation within six months and pregnant within six months or cyst size less than 3 cm in the sixth month). Therefore, when the patient has recovered, the value of this attribute is set to 1; otherwise, it is set to 0. Moreover, Cyst_size, CA_125, and BMI are the size of cyst, blood test value, and body mass index before operation, respectively. The resultant rules generated from the decision tree are shown as follows:

(1) If Cyst_size ≥ 4.25 Then Recovery = 0 (recovery rate =32%)

(2) If Cyst_size < 4.25 and CA_125 ≥ 146.81Then Recovery = 0 (recovery rate =0%)

(3) If Cyst_size < 4.25 and CA_125 < 146.81and BMI < 22.7516 Then Recovery = 1 (recovery rate =80.77%)

(4) If Cyst_size < 4.25 and CA_125 < 146.81and BMI ≥ 22.7516 Then Recovery = 0 (recovery rate =25%)

The CART is a binary tree, as shown in figure 1, there is a significant difference between two branches of a node when the pruning process is performed automatically by the mining tool. Therefore, we can interpret the corresponding rules from the tree directly. However, to strengthen our conclusion, we still divided all the patients are into two different groups according to the corresponding rules obtained from the tree. The significance level is 0.05. The recovery rate of these 212 patients is 43.4%.

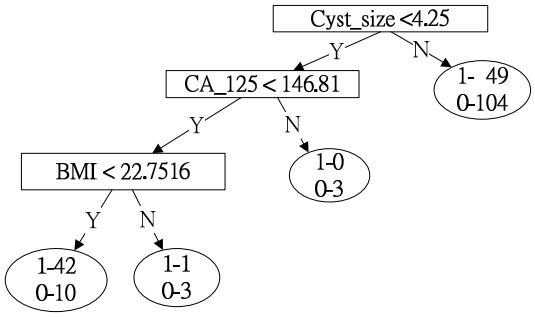


Figure 1. The resultant decision tree.

As shown by test 1 in Table II, according to the rule "Cyst_size ≥ 4.25 ==> Recovery = 0" all patients are divided into two groups by the cut point of the Cyst_size attribute. Group 1 contains patients whose cyst size is less than 4.25, while cyst size of patients in group 2 are not less than 4.25. From test 1 in the table, we conclude that when all patients are divided into two groups: group 1 (Cyst_size <4.25) and group 2 (Cyst_size ≥ 4.25). The recovery rate of group 1 (43/59, 72.88%) is significantly (p = 1.47E-08) greater than that of group 2 (49/153, 32%). In other words, the recovery rate (or recurrence rate) is affected by the cyst size regardless of the treatment type. Moreover, from test 2 and 3 in Table Ⅱ, we also conclude that the recovery rate of group 1 is significantly better than that of group 2.

Since each branch of the decision tree is only represented by a rule, and the resultant rules of the tree are local [27], some useful description rules cannot be found by the tree-generation algorithm. For example, the rule "Cyst_size < 4.25 and CA_125 ≥ 146.81 ==> Recovery = 0" should be ignored in the discussion because only three of the patients are related to this rule and the over-fit problem. Consequently, the question is "for those patients whose Cyst_size < 4.25, whether the BMI value will affect the recovery rate." Since our data size is very small and there are only three cut points in this example, to overcome the above problems, we could use these three cut points to generate all possible rules from the tree. We discuss this phenomenon in the

TABLE II.
COMPARISON OF DECISION TREE ALGORITHMS

test	Group 1	Group 2	Group 1 Recovery-rate (y/total, rate)	Group 2 Recovery-rate (y/total, rate)	*p*-value
1	Cyst_size <4.25	Cyst_size≥4.25	43/59. 72.88%	49/153, 32%	1.47E-08
2	Cyst_size <4.25 and CA_125 <146.81	Cyst_size <4.25 and CA_125 ≥146.81	41/53, 77.36%	0/3, 0%	0.001354
3	Cyst_size <4.25 and CA_125 <146.81 and BMI< 22.7516	Cyst_size <4.25 and CA_125 <146.81 and BMI≥22.7516	36/43, 83.72%	1/4, 25%	0.002647

following section.

Ⅳ. Interpreting Mining Results Regardless to Treatment Type

From the tree in figure 1, we obtain the cut points of Cyst_size, CA_125, and BMI numeric attributes as 4.25, 146.81, and 22.752, respectively. We can use these three values to generate all possible rules from the tree. Because the data set only contains 212 records, it is small. So, we used all of these data, as in training data. However, there are different conditions for generating all possible rules that could generate different recovery rates for the patients. Therefore, using single rule information, we cannot filter useful rules from all possible rules. And the study case aim is to investigate recurrent ovarian endometriomas under different conditions in a possible rule. For these reasons, we use *t*-test to verify rules to discover some useful description rules after all possible rules from the tree are generated. Then, we use *t*-test to verify all possible rules obtained from the decision tree to discover useful knowledge. Moreover, since the values of BMI and CA_125 may be missing, the number of patients of BMI or CA_125 attribute is not equal to 212.

A. Effect of CA_125 Value

The rule "Cyst_size < 4.25 and CA_125 ≥ 146.81 ==> Recovery = 0" should be ignored in the discussion because of the over-fit problem. However, there are 13 patients whose CA_125 value is greater than 146.81 in the whole training data. Therefore, we should check whether this CA_125 value presents some interesting descriptions about the recovery rate.

In this section, all patients are divided into two groups according to three different conditions, as shown in Table III. As shown by test 1 in Table III, all patients are divided into two groups according to the cut point of the CA_125 attribute. Group 1 contains patients whose CA_125 values are less than 146.81, while CA_125 values of patients in group 2 are not less than 146.81. From test 1 in the table, we conclude that when all patients are divided into two groups, the recovery rate of group 1 (89/189, 47.09%) and group 2 (1/13, 7.69%) is significantly different (p = 0.002769). From tests 2 and 3, we conclude that when a patient's CA_125≥146.81, there is no significant difference between the recovery rates of the two groups. In other words, when CA_125 ≥ 146.81, the recovery rate (or recurrence rate) will mainly be affected by this CA_125 value regardless of the cyst size and BMI value. That is, the recovery rate of that patient would be very low regardless of treatment type. These new useful descriptions cannot be discovered by the tree-generation algorithm directly. Actually, when a patient's CA_125 ≥ 146.81, the patient must have other diseases; otherwise, the value would not be greater than or equal to 146.81.

B. Effect of Cyst_size and BMI Values

As indicated in the above section, when CA_125 ≥ 146.81, the recovery rate (or recurrence rate) will mainly be affected by this CA_125 value regardless of the Cyst_size and BMI values. Therefore, we do not consider this attribute in the following discussion. According the cut points of the Cyst_size and BMI attributes, as shown in Table Ⅳ, all patients can be divided into two groups according to four different conditions. From Table Ⅳ, we conclude that the recovery rate (or recurrence rate) will mainly be affected by the cyst size.

TABLE IV.
RECOVERY RATES OF GROUPS

test	Group 1	Group 2	Group 1 Recovery-rate (y/total, rate)	Group 2 Recovery-rate (y/total, rate)	*p*-value
1	Cyst_size <4.25	Cyst_size ≥4.25	43/59. 72.88%	49/153, 32%	1.47E-08
2	BMI< 22.7516	BMI≥22.7516	75/155, 48.39%	9/26, 34.62%	0.097306
3	Cyst_size <4.25 and BMI< 22.7516	Cyst_size <4.25 and BMI≥22.7516	37/47, 78.72%	1/5, 20%	0.002115
4	Cyst_size ≥4.25 and BMI< 22.7516	Cyst_size ≥4.25 and BMI≥22.7516	38/108. 35.19%	8/21, 38.1%	0.400407

TABLE III.
T-TEST RESULTS FOR CA_125

test	Group 1	Group 2	Group 1 Recovery-rate (y/total, rate)	Group 2 Recovery-rate (y/total, rate)	*p*-value
1	CA_125<146.81	CA_125≥146.81	89/189, 47.09%	1/13, 7.69%	0.002769
2	CA_125≥146.81 and Cyst_size<4.25	CA_125≥146.81 and Cyst_size≥4.25	0/3, 0%	1/10, 10%	0.302958
3	CA_125≥146.81 and BMI< 22.7516	CA_125≥146.81 and BMI≥22.7516	1/8, 12.50%	0/4, 0%	0.252925

From test 2 in Table II, we conclude that, for patients whose cyst size is less than 4.25 and CA_125 is not less than 146.81, the BMI value, 22.7516, will significantly affect the recovery rate. However, we already know that when a patient's CA_125 is greater than 146.81, the recovery rate of the patient is very low. Therefore, the question in mind is that how about for those patients whose cyst size is less than 4.25. Will the value of BMI significantly affect the recovery rate of patients regardless of the CA_125 value? From test 3 in Table Ⅳ, we conclude that the recovery rates of group 1 (43/59, 72.88%) and group 2 (1/5, 20%) are significantly different (p = 0.002115). In other words, we can say that when the cyst size is less than 4.25, the BMI value will affect the recovery rate. We get a new useful description that cannot be discovered by the tree-generation algorithm directly. This new description can be interpreted as follows: "For those patients whose cyst size is less than 4.25, when the patient's BMI value is less than 22.7516, the recovery rate is 78.72%; otherwise, the recovery rate is 20% only. That is, for those patients whose cyst size is less than 4.25, when the BMI value is not less than 22.7516, the value of BMI will significantly affect the recovery rate."

C. Interpreting Mining Results with Ethanol Instillation

As pointed out by Noma and Yoshida [9], ethanol instillation into the cyst cavity for more than 10 min was most effective at reducing the recurrence rate. Therefore, another goal is to verify whether aspiration followed by the injection of 95% ethanol left in situ is an effective treatment for ovarian endometriomas under different conditions. Therefore, in the data preprocessing step, we classify the values of ethanol irrigation duration into two types: less than 10 min and retention. Therefore, for each condition, all patients are divided into two groups (group 1: ethanol irrigation; group 2: ethanol retention) by the resultant cut points. The values of *t*-test for the above conditions are shown in Table V. From the table, we give the following conclusions:

(1) When CA_125 > 146.81, recurrence rate is very high regardless of treatment type.

(2) When Cyst_size ≥ 4.25, the recovery rate of group 1 (11/49, 22.45%) and group 2 (38/104, 36.54%) is significantly different (p = 0.041180).

(3) Except the above two conditions, although the recovery rate of group 2 is better than that of group 1, there is no significant difference between these two groups.

TABLE V.
RECOVERY RATES OF ETHANOL IRRIGATION AND ETHANOL RETENTION GROUPS ABOUT DIFFERENT CONDITIONS

Condition	Group 1 Recovery-rate (y/total, rate)	Group 2 Recovery-rate (y/total, rate)	p-value
All patients	27/83, 36.99%	65/139, 46.76%	0.086980
CA_125 <146.81	24/59, 40.68%	65/130, 50%	0.118197
CA_125 ≥146.81	1/5, 16.67%	0/7, 0%	0.149845
Cyst_size <4.25	16/24, 66.67%	27/35, 77.14%	0.191314
Cyst_size ≥4.25	11/49, 22.45%	38/104, 36.54%	0.041180
BMI < 22.7516	21/46, 45.65%	54/109, 49.54%	0.330266
BMI≥22.7516	3/12, 25%	6/14, 42.86%	0.180010
Cyst_size <4.25 And BMI< 22.7516	12/17, 70.59%	25/30, 83.33%	0.157719
Cyst_size <4.25 And BMI≥22.7516	1/3, 33.33%	0/2, 0%	0.247512
Cyst_size ≥4.25 And BMI< 22.7516	9/29, 31.03%	29/79, 36.71%	0.294158
Cyst_size ≥4.25 And BMI≥22.7516	2/9, 22.22%	6/12, 50%	0.106864

Ⅴ. DISCUSSION AND CONCLUSION

Endometriosis is a complex disease with several attributes. Each of the attributes, such as the stage of the disease, symptoms of the disease, serum level of CA-125, can affect the making of treatment decisions and the results of the treatment.

This study integrates the statistical method and decision tree techniques to analyze the postoperative status of ovarian endometriosis patients. Experimental results show that some new interesting knowledge can be found by our approach. However, our method works only for a simple tree. To deal with more complex decision trees, we want to integrate other data mining the approaches to analyze postoperative status of ovarian endometriosis patients in the future. We hope to find more precise description knowledge about the recovery rates of the patients under different conditions.

REFERENCES

[1] . Tang, T.I., et al., "A Comparative Study of Medical Data Classification Methods Based on Decision Tree and System Reconstruction Analysis," Industrial Engineering and Management Systems, vol. 4, p. 102¡V108, 2005.

[2] . Cui, Z. and G. Zhang, "A novel medical image dynamic fuzzy classification model based on ridgelet transform," Journal of Software, vol. 5, pp. 458-465, 2010.

[3] . Scheetz, L.J., J. Zhang, and J. Kolassa, "Classification tree modeling to identify severe and moderate vehicular injuries in young and middle-aged adults," Artificial Intelligence in Medicine, vol. 45, pp. 1-10, 2009.

[4] . Ozkan, S. and A. Arici, "Advances in treatment options of endometriosis," Gynecol Obstet Invest, vol. 67, pp. 81-91, 2009.

[5] . Aboulghar, M.A., et al., "Ultrasonic transvaginal aspiration of endometriotic cysts: an optional line of treatment in selected cases of endometriosis," Human Reproduction, vol. 6, p. 1408, 1991.

[6] . Aboulghar, M.A., et al., "Treatment of recurrent chocolate cysts by transvaginal aspiration and

tetracycline sclerotherapy," Journal of assisted reproduction and genetics, vol. 10, pp. 531-533, 1993.
[7] . Mesogitis, S., et al., "Combined ultrasonographically guided drainage and methotrexate administration for treatment of endometriotic cysts," The Lancet, vol. 355, pp. 1160-1160, 2000.
[8] . Acien, P., et al., "GnRH analogues, transvaginal ultrasound-guided drainage and intracystic injection of recombinant interleukin-2 in the treatment of endometriosis," Gynecologic and obstetric investigation, vol. 55, pp. 96-104, 2000.
[9] . Noma, J. and N. Yoshida, "Efficacy of ethanol sclerotherapy for ovarian endometriomas," International Journal of Gynecology & Obstetrics, vol. 72, pp. 35-39, 2001.
[10] . Akamatsu, N., et al., "Ultrasonically Guided Puncture of Endometrial Cyst: Aspiration of Contents and Infusion of Ethanol," Acta Obstetrica et Gynaecologica Japonica, vol. 40, pp. 1214-1215, 1988.
[11] . Hsieh, C.L., et al., "Effectiveness of ultrasound-guided aspiration and sclerotherapy with 95% ethanol for treatment of recurrent ovarian endometriomas," Fertility and sterility, vol. 91, pp. 2709-2713, 2009.
[12] . Akan, E., et al., "Predictive Power of Activin A Levels in the Prognosis of First Trimester In Vitro Fertilization Pregnancies," Journal of Women's Health, 2011.
[13] . Acien, P., et al., "Use of intraperitoneal interferon [alpha]-2b therapy after conservative surgery for endometriosis and postoperative medical treatment with depot gonadotropin-releasing hormone analog: a randomized clinical trial* 1," Fertility and sterility, vol. 78, pp. 705-711, 2002.
[14] . Takano, M., et al., "The impact of complete surgical staging upon survival in early-stage ovarian clear cell carcinoma a multi-institutional retrospective study," International Journal of Gynecological Cancer, vol. 19, pp. 1353-1357, 2009.
[15] . Van Calster, B., et al., "Towards a clinical decision support system for pregnancies of unknown location," in *Proceedings - IEEE Symposium on Computer-Based Medical Systems*, Jyvaskyla, 2008, pp. 581-583.
[16] . Ben VAN CALSTE, "Predictive diagnostic models for gynecologic applications with focus on multi-class classification," PhD thesis, Dept. of Electrical Engineering, Katholieke Universiteit Leuven, Belgium, 2008.
[17] . Kyama, C.M., et al., "Evaluation of endometrial biomarkers for semi-invasive diagnosis of endometriosis," Fertility and sterility, vol. 95, pp. 1338-1343.e3, 2011.
[18] . Seeber, B., et al., "Panel of markers can accurately predict endometriosis in a subset of patients," Fertility and sterility, vol. 89, pp. 1073-1081, 2008.
[19] . Liu, H., et al., "Detection of endometriosis with the use of plasma protein profiling by surface-enhanced laser desorption/ionization time-of-flight mass spectrometry," Fertility and sterility, vol. 87, pp. 988-990, 2007.
[20] . Van Holsbeke, C., et al., "Endometriomas: Their ultrasound characteristics," Ultrasound in Obstetrics and Gynecology, vol. 35, pp. 730-740, 2010.
[21] . Bellazzi, R. and B. Zupan, "Predictive data mining in clinical medicine: Current issues and guidelines," international journal of medical informatics, vol. 77, pp. 81-97, 2008.
[22] . Kim, I.C. and Y.G. Jung, "Using Bayesian networks to analyze medical data," in *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, Leipzig, 2003, pp. 317-327.
[23] . Hsieh, C.L., C.S. Shiau, and L.M. Lo, "Effectiveness of ultrasound-guided aspiration and sclerotherapy with 95% ethanol for treatment of recurrent ovarian endometriomas," Fertility and sterility, vol. 91, pp. 2709-2713, 2009.
[24] . Kinkel, K., et al., "Indeterminate ovarian mass at US: Incremental value of second imaging test for characterization-meta-analysis and Bayesian analysis," Radiology, vol. 236, pp. 85-94, 2005.
[25] . Wang, H. and P. Zhang, "A quantitative method for pulse strength classification based on decision tree," Journal of Software, vol. 4, pp. 323-330, 2009.
[26] . Breiman, L., et al., "Classi cation and regression trees," Wadsworth, Belmont, 1984.
[27] . Wang, K., S. Zhou, and Y. He, "Growing decision trees on support-less association rules," in *Proceeding of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA, 2000, pp. 265-269.

Ming-Yang Chang, M.D. is currently working as an Associated professor in Gynecology, Chang Gung University School of Medicine, Taoyuan, Taiwan. His research interests include Infertility, endometriosis and female reproductive medicine.

Chien-Chou Shih received the Ph.D. degree in information engineering from Tamkang University, Taiwan, in 1998. He is currently an Assistant Professor with the Department of Information and Communication, Tamkang University. His research interests include embedded software programming, data mining applications, and engineering design education.

Dina-An Chiang received the BS degree in hydraulic engineering from Chung Yuan Christian University, Taiwan, in 1981, and the MS and PhD degrees in computer science from the University of Southwestern Louisiana in 1986 and 1990, respectively. He is currently a professor in the Department of Computer Science. His research interests include fuzzy, relational databases and data mining.

Chun-Chi Chen received the MS degrees in computer science and information engineering from Tamkang University in Taipei, Taiwan, in 2003. His research interests include relational databases and data mining.

Improvement of Filtering Algorithm for RFID Middleware Using KDB-tree Query Index

Xiaobo Zhang
Faculty of Automation, Guangdong University of Technology, Guangzhou 510006, China
Email: zxb_leng@163.com

Lianglun Cheng
Faculty of Automation, Guangdong University of Technology, Guangzhou 510006, China
Email: lcheng@gdut.edu.cn

Quanmin Zhu
Intelligent Autonomous Systems Lab, University of the West of England, Bristol,
Coldharbour Lane, Bristol BS16 1QY, UK
Email:QuanZhu@uwe.ac.uk

***Abstract*—RFID middleware collects and filters RFID streaming data gathered continuously by numerous readers to process requests from applications. These requests are called continuous queries. The problem when using any of the existing query indexes on these continuous queries is that it takes a long time to build the index because it is necessary to insert a large number of segments into the index. KDB-tree is an index which can dispose multidimensional data. It is also a dynamic balance tree that has a good query performance and high spatial usage. This paper propose an aggregate transformation algorithm for querydata filtering, and applies KDB-tree into RFID event filtering to improve the performance of query. Comparing to other indexes, the result of simulation shows that KDB-tree index outperforms others in synthesized consideration of storage cost, insertion time cost and query time cost. In particular the query time cost of KDB-tree is distinctly lower than others because it provides single path traverse in the query process.**

***Index Terms*—RFID middleware; filtering algorithm; KDB-tree; querydata; pointquery**

I. INTRODUCTION

RFID middleware collects and filters the data transferred by readers. This work will bring high load for RFID system with two reason, one is the reader's continual action will produce continual data, the other is multiple ECSpec prescribes multiple query processes for RFID events and these query processes could make RFID middleware continually filter tag data according to multiple conditions. For these reasons, RFID middleware must have a good algorithm in order to filter events from tag data quickly and accurately.

To present, the researches about event filtering of RFID middleware mostly focused on the theory of memory database[1-3]. The method is to search better index to improve the query process. A better way for improvement of the query process is to construct various multidimensional indexes[4-5]. There are many indexes could be applied into RFID event filtering such as Hash,CQI,VCR,R-tree etc. CQI(*Cell-based Query Index*) is a kind of index on the basis of memory[6]. In such index, query scope is divided into various cells. Every cell has a query table which intersects or joins query conditions, the cell uses this query table to filter results. VCR(*Virtual Construct Rectangle*) index is also a kind of index on the basis of memory[7]. In such index, query scope is divided into multiple segments. These segments are encapsulated by VCs(*virtual constructs*) with inserted ID. We can search these VCs for results. CQI and VCR both are two-dimensional space composed of RID(*Reader Identification Domain)* and TID(*Tag Identification Domain)* ,and both regard constant query as a whole domain, while ECSpec is not a domain but segment. If such indexes were applied into RFID query, there must be enough memory for distribution and the times of insertion will be high. An improvement method is to use multidimensional indexes such as R-tree [8-10]. While in such index, there are four-dimensional data should be stored: RID, manager, product and serial number [11]. For this reason, the performance of this index will decline for high dimensions. As we know, the query performance of the index deteriorates exponentially with higher dimensions [12]. Thus, the traditional indexes can decrease many insertions of querydata, but they are not efficient for query.

Due to the disadvantages of these multidimensional indexes, this paper provides a new index like KDB-tree for RFID middleware [13]. KDB-tree is also a kind of

This work was supported in part by the Guangdong Natural Science Foundation(#8351009001000002 and #07117421), the Major Program of Guangdong Natural Science Foundation(#2009A080207008), the Joint Funds of the National Natural Science Foundation of China and Guangdong Natural Science Foundation(#U0935002) , the Youth Foundation of Guangdong University of Technology(#405095245).
Corresponding author.: zxb_leng@163.com (Xiaobo Zhang)

doi:10.4304/jsw.6.12.2521-2527

multidimensional index composed of RID, SID and TID, but it can decrease the number of insertion and the space cost, and also keep the tree in balance. Besides these, in KDB-tree index, a pointquery has only one path from root to leaf.

II. AGGREGATION AND CONVERSION OF QUERYDATA

Definition 1*: RFID middleware collects and filters a tagdata from reader for once. This process is called a pointquery, the expression can be described as PointQuery (RIDi, TIDj).*

Definition 2*: querydata is a continual query based on ECSpec and is a filtering condition for reader and tag. RID is the scope of readers and TID is the filtering condition for tags.*

Definition 3*: querydata with only one segment is called simple querydata and querydata with two or more segments is called complicated querydata.*

Querydata includes one or more segments which are composed of RID and TID in two-dimensional space. A segment is the smallest unit in complicated querydata. if querydata is set to d (d={d1,…,dn}, $1<=n<=2^{24}$), then the segment of querydata could be set to di={(minrid,mintid),(maxrid,maxtid)}, di∈d.

In two-dimensional space of (RID, TID), the querydata is composed of discrete segments. If user wants to search “the Nokia cellphones made in this year in warehouse A”, the system will send an ECSpec request including CQ1: “readerID=2,EPC-Pattern=<10.[1-3].[3001-4000]>”[14]. It shows that the readerID in warehouse A is 2, CQ1 arrives RFID middleware and is inserted into query index, then the querydata has three discrete segments, the first is {(1, 10·260+ 1·236 + 3001), (1, 10·260+ 1·236+ 4000)}, the second is {(1, 10·260+ 2·236+ 3001), (1, 10·260+ 2·236+ 4000)}, and the last is {(1, 10·260+ 3·236+ 3001), (1, 10·260+ 3·236+ 4000)}.

From above we can see that if a product ID P with EPC mode is a scope, the querydata will include multiple segments. The size of the segment is determined by the scope of serial number. Querydata is a complicated object which includes 2^{24} segments at most. When a continual ECSpec query is executing, we must insert segments into index. Too many times of insertion will make index much larger and the performance of pointquery will decline.

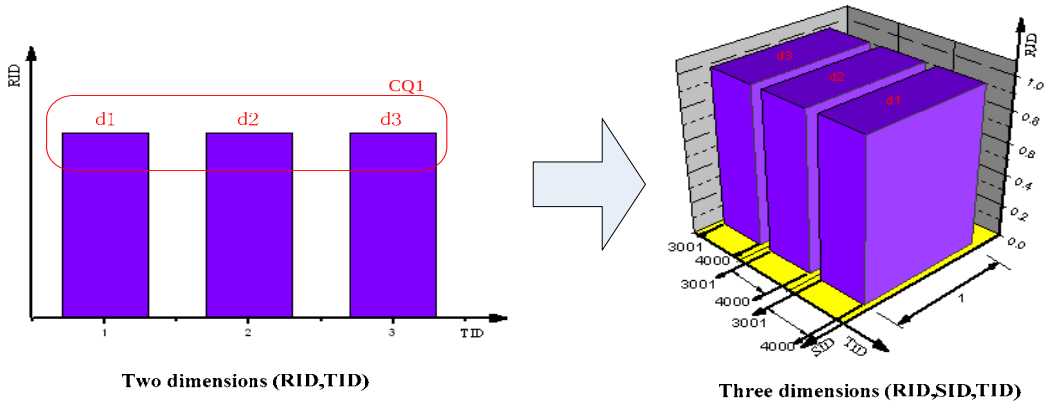
To reduce the insertion quantity, the number of segments must be declined. This paper provides an aggregation and conversion method that transform two dimensional data (RID,TID) to three dimensional data (RID,SID,TID). ECSpec provides 27 different modes while only 11 modes have actual significance. The querydata is determined by EPC mode which includes Manager, Serial Number and Product. In EPC mode, every part is a constant number, or [low-high] ,or *.

Definition 4*: there are three dimensionalities:(s,t are two dimensionalities in two-dimensional space.)*

S: $m*2^{24}+s$;
R:RID;
T:t

Then any querydata in this three-dimensional space can be expressed as one segment: d={(minRID,minSID,minTID),(maxRID,maxSID,maxTID)}.

Every querydata only need to be transformed into a three-dimensional query segment by RFID middleware (see in Fig.1.). In this way, the quantity of insertion could be declined.



(CQ1: readerID=2,EPC-Pattern=<10.[1-3].[3001-4000]>)
Fig.1. aggregation and conversion of querydata

III. FILTERING ALGORITHMS OF KDB-TREE

A. Data Structure of KDB-tree Index

Like B-tree, KDB-tree consists of a collection of pages and a variable root ID that gives the page ID of the root page. There are two types of pages in a KDB-tree:

1)Region pages: region pages contain a collection of pairs(*region, page ID*).

2)Point pages: point pages contain a collection of pairs(*point, location*), in which *location* gives the location of a database record. The pair (*point, location*) is an index record.

KDB-tree has following properties:

1)Considering each page as a node and each page ID in a region page as a node pointer, the resulting graph structure is a tree with root *root ID*. Furthermore, no region page contains a null pointer, and no region page is

empty.

2)The path length, from root page to leaf page is the same for all leaf pages. That is, the KDB-tree is a balance tree.

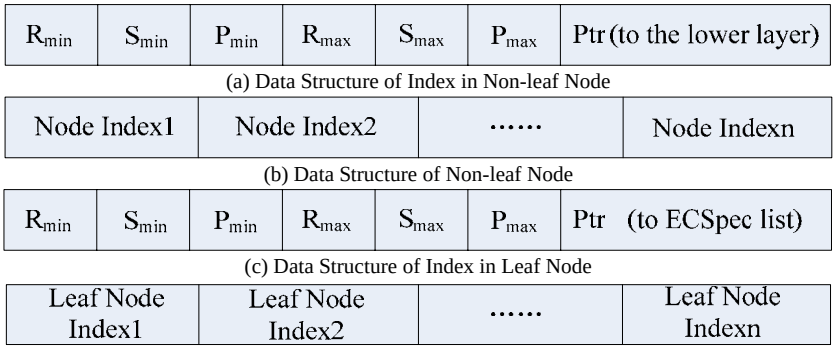
3)In every region page, the regions in the page are disjoint, and their union is a region.

4)If the root page is a region page, the union of its regions is *domain0*×*domain1*×…×*domainK-1*.

5)If (*region,child ID*) occurs in a region page, and the child page referred to by *child ID* is a region page, then the union of the regions in the child page is *region*.

6)Referring to (5), if the child page is a point page, then all the points in the page must be in *region*.

In RFID middleware, KDB-tree index is a three-dimensional structure with the three-dimensional data aggregated and conversed from two-dimensional data, including non-leaf node structure and leaf node structure. The items of non-leaf node store three dimensional information {(Rmin，Smin，Pmin)，(Rmax，Smax,Pmax)} and a pointer to the lower layer, while the items of leaf node store the query scope of ECSpec {(Rmin，Smin，Pmin)，(Rmax，Smax,Pmax)} and a pointer to the ECSpec list. As is shown in Fig.2.



(a) Data Structure of Index in Non-leaf Node

(b) Data Structure of Non-leaf Node

(c) Data Structure of Index in Leaf Node

(d) Data Structure of Leaf Node

Fig.2. data structure of KDB-tree index

B. Pointquery Algorithm of KDB-tree Index

KDB-tree is a kind of multidimensional index, and has a higher query performance, especially in pointquery. KDB-tree provides a kind of traverse with single path. Every pointquery has one path from root to leaf, and can always keep the tree balance in dynamic insertion.

While RFID middleware collects a tagdata from RFID reader and transforms it into three-dimensional data (RID,SID,TID), then searches the suitable querydata in KDB-tree index. This process is called a pointquery. The process can be described with function PointQuery(root,d) in which d is a querydata. Following is the pointquery algorithm:

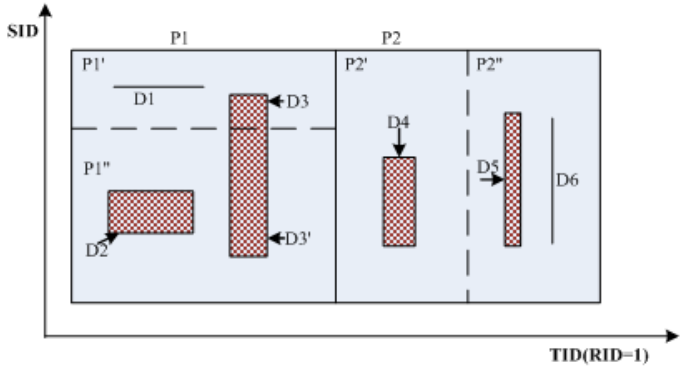
Algorithm 1: PointQuery(root,d)

```
PointQuery(root,d)
{
  If root is NULL return FALSE;
    for each entry root.e {
      If Contain(root.e,p) is TRUE  {
        If root is a leaf     Return TRUE;
        If root is not a leaf   PointQuery(e,d);
      }
    }
}
```

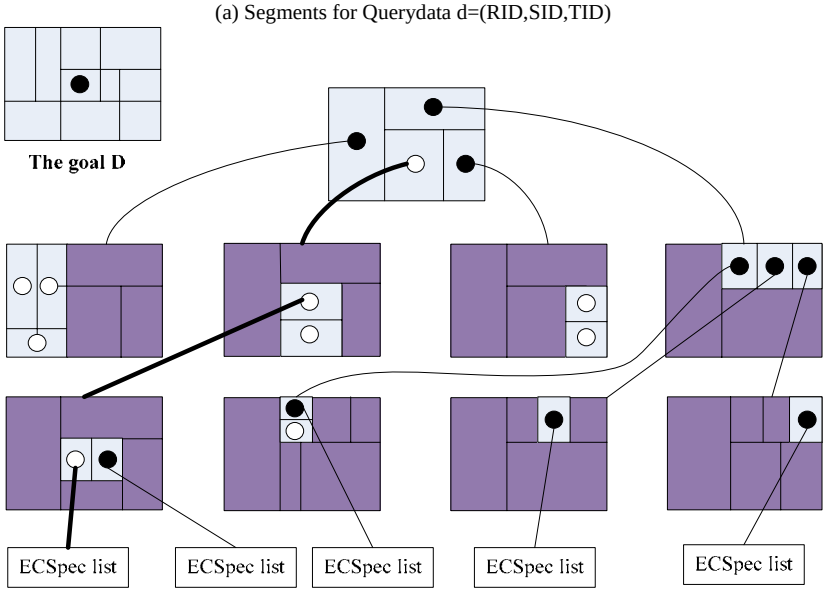
Pointquery starts from root. If the visited node is NULL, it shows that pointquery did not find the querydata segment with d, then it traverses every node in prior order, transfer *Contain(e,p)* for every segment *e*. If the result is TRUE, pointquery will go on judging whether it is a leaf node. If it is a leaf node, then it indicates that the pointquery has found the querydata segment with *d*, otherwise recursivly transfer the pointquery procedure.

Fig.3 is an example of pointquery. In Fig.3(a), the rectangle represents every querydata segment. Every querydata segment could be displayed in two-dimensional space due to RID=1. Then RFID middleware transfers the function *transform()* to converse (RID,TID) to d=(RID,SID,TID) before pointquery procedure.

In Fig.3(b), the rectangle in the left top expresses the position of the goal D in KDB-tree index and its space scope. Every other rectangle expresses an index item, the size and position of the rectangle is the space scope and the position in the KDB-tree index. Several index items compose to a node, like the colored parts in the rectangle. In leaf node, the rectangle with a dot expresses the scope of the querydata. The bold line is the path of the goal D in traversing KDB-tree. When in traversing, we should start from root node, find the index item with the scope of goal D and visit the next node by the pointer in index item. Repeat this process till we find the index item with the scope of goal D in leaf node. Then we can visit ECSpec list by the pointer in this index item and the pointquery ends.



(a) Segments for Querydata d=(RID,SID,TID)



(b) Process of Pointquery

Fig.3. Example of Pointquery

C. Insertion Algorithm of KDB-tree Index

When RFID middleware accepts an ECSpec request, it converses the querydata into a node N, as is shown in the left top in Fig.4. The insertion procedure is composed of two steps, the transformation step and the insertion step. Before querydata is inserted into the query index, it should firstly be converted into aggregated data. This process is executed by the transform function. After conversion, the insertion process of the aggregated data is the same as the original KDB-tree algorithm. The insertion algorithm of KDB-tree index could be described as following:

Algorithm 2:Insertion Algorithm

①Firstly traverse the index from the root node with single path till a leaf node LN.

②If LN has already had an index item, and its scope is the node N's scope, then a new ECSpec1 list should be added to the existing ECSpec list.

③Otherwise find an index item which scope includes the node N's scope and execute the split algorithm.

④Adjust the scope of the index item and add a new ECSpec1 list to its existing ECSpec list.

⑤Create a new node newLN which scope equals the node N's scope, and create a new ECSpec list newEL and insert to ECSpec1.

⑥make newLN point to newEL.

The split strategy is based on two aspects. The first is to determine an efficient split axis in (RID,SID,TID). The complexities of the axes could be calculated in the node. The complexity of an axis is the ratio of data to the space of the axis when the data in the node are projected onto the axis. The complexity of an axis is a value dividing the summation of lengths of data in the node by the entire length of the node on the axis. We choose the axis that has the minimum complexity value as the split axis because an uncomplicated axis minimizes the number of duplications of data.

The second is to determine an effective split position on the split axis. It considers dispersion and duplication. The degree of dispersion is the distribution ratio of data in the overflow node. The degree of duplication is the amount of data stored in both sides of the split. The equation that finds the split position by accommodating these two factors is as following.

$$SP_i = e \cdot D_1 + (1-e) \cdot \left(\frac{1}{2}\right)^{D_2} \quad (0 \le e \le 1) \tag{1}$$

SPi is the *ith* split position on the split axis, *D1* is the distributed ratio of data in the overflow node, *D2* is the number of data stored on both sides of the split, and *e* is a constant value to give more weight between the two factors. According to Equation 1, the SP value increases

as the degree of dispersion (*D1*) increases and the degree of duplication (*D2*) decreases. We choose the maximum *SPi* from the values calculated by Equation 1. Then, the data in the overflow node are distributed more uniformly and duplicated less by the split. Following is the split algorithm:

Algorithm 3: Split(N_{old}, N_{new}, obj)

axis = FindSplitAxis(N_{old}, obj);

sp = FindSplitPosition(N_{old}, obj, axis);

if N_{old} is leaf node

store obj into sp on axis in N_{new} and delete them in N_{old};

else{

store MBBs into sp on axis in N_{new} and delete them in N_{old};

add the entry pointing N_{new} to parent node of N_{old};

}

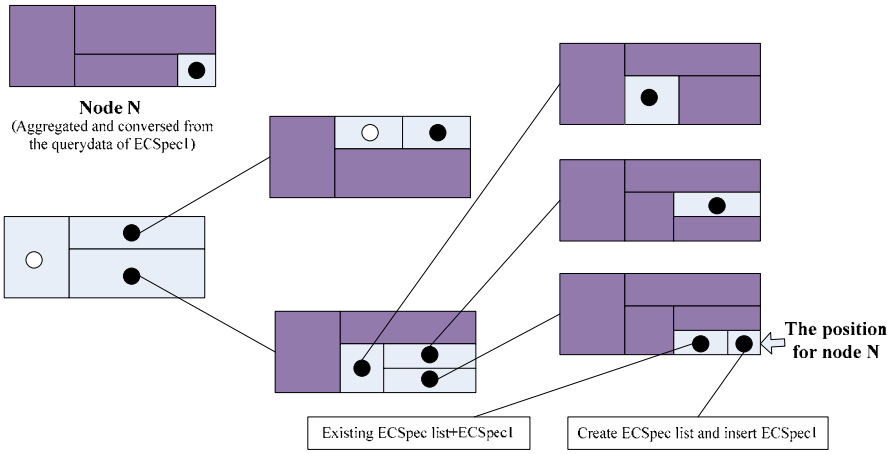


Fig.4. Insertion process of KDB-tree

IV. Simulation and Performance Comparison

In this simulation, we do a comparison with KDB-tree、R-tree、CQI and VCR index. KDB-tree index is based on three-dimensional space (RID,SID,TID), R-tree index is based on four-dimensional space (RID, Manager, Product, Serial number), while CQI and VCR are based on two-dimensional space (RID, TID). These indexes are stored in memory and use *Wall Clock Time* as measurement. The simulation of these indexes use the same ECSpec data and pointquery data. The comparison of these indexes is in three aspects: storage costs, insertion time costs and comprehensive performance.

The simulation platform of RFID event filtering consist of virtual reader, virtual client and RFID middleware. To reduce the effect of the virtual reader and virtual client on RFID event filtering, the structure of system is distributed mode. That is, virtual reader, virtual client and RFID middleware runs in different computers in LAN. The communication between virtual reader(virtual client) and console platform is implemented by web service.

The first comparison is the storage costs of KDB-tree、R-tree、CQI and VCR by using 5000, 10000, 50000, 100000 querydata to insert, as is shown in Fig.5. The result shows that CQI and VCR need the largest storage for the querydata must be separated into multiple segments to be inserted into index. R-tree needs the smallest storage. It transforms a querydata into four-dimensional data and inserts the data into index only once. KDB-tree needs more storage than R-tree, and transforms a querydata into three-dimensional data.

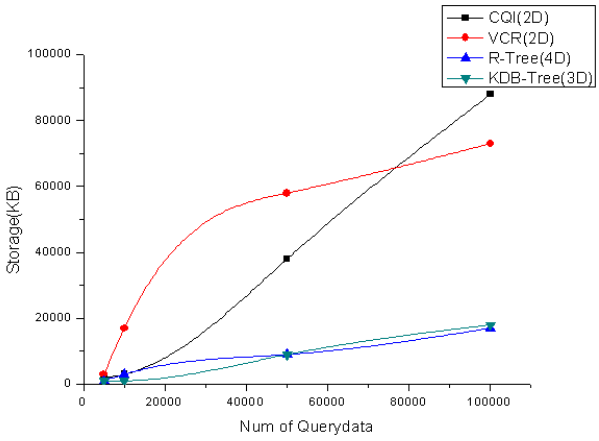


Fig.5 comparison of storage costs

The second comparison is the insertion time costs of KDB-tree、R-tree、CQI and VCR by using 5000, 10000, 50000, 100000 querydata to insert, as is shown in Fig.6. The insertion time for CQI is lowest, whereas the insertion time for VCR is highest for the querydata must be separated into multiple segments. KDB-tree and R-tree have the same time costs for insertion.

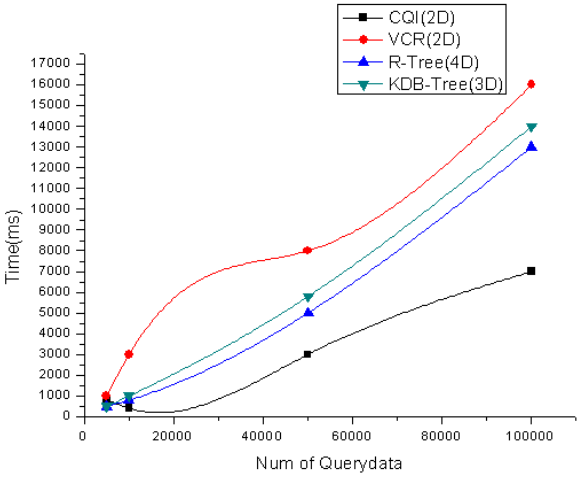


Fig.6 comparison of insertion time costs

The last comparison is the comprehensive performance, as is shown in Fig.7. The comprehensive performance of CQI and VCR is lowest for they need more time to insert than other indexes. KDB-tree's comprehensive performance is better than R-tree's for KDB-tree could traverse every node by single path while R-tree needs multiple paths to traverse. The result of the simulation shows that KDB-tree is the best query index in the index family.

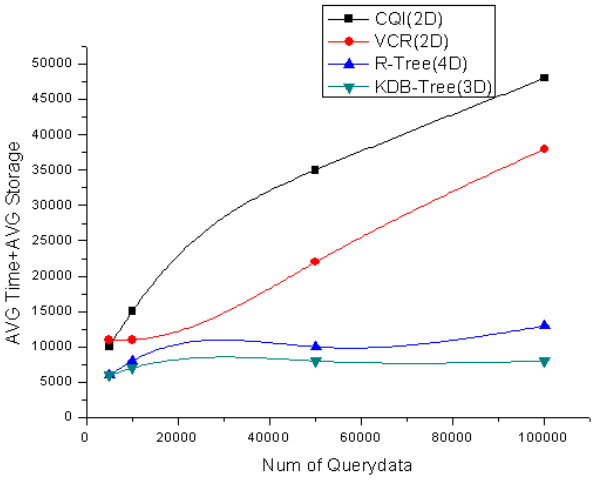


Fig.7 comparison of query performance

V. CONCLUSION

The query index approach is usually suitable for evaluating continuous queries over streaming data. Query indexes in previous studies have treated a simple query as the data, such as a region continuous query or an interval continuous query. However, the data in the RFID query index is represented as a great many segments because it is a continuous query from the ECSpec. With any of the existing methods as the RFID query index, it is very time consuming to build an index on these data, because it is necessary to insert a great many segments into the index for storing a continuous query. The index size also makes it inefficient to process queries.

The paper provides an aggregation and conversion method to querydata and a KDB-tree index for RFID middleware. There are several indexes for RFID event filtering, but they also have their shortages in some aspects. The result of simulation shows that KDB-tree has a comparatively good performance than other indexes. The algorithm based on KDB-tree index could be applied into actual RFID system to make RFID middleware filter large tagdata more quickly and accurately. So the performance of RFID system could be improved.

ACKNOWLEDGMENT

We would like to express our great appreciation to all the members of our laboratory for their technical insights and stimulating ideas, which greatly contributed to the success of our research.

REFERENCES

[1] YU Jian,LAI Sheng-li. Structure of main memory databases of radio frequency identification middleware. *Journal of Harbin Engineering University*.2008,Vol. 29,no. 6, pp.578-582

[2] Rong-Jhang Liao, Pei-Lun Suei, Yung-Feng Lu, Tei-Wei Kuo, Chun-Sho Lee. A Signature-based Grid Index Design for RFID Main-Memory Databases. *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2008. EUC '08.* Vol. 1, pp.519-525

[3] Bin Cui, Beng Chin Coi, Jianwen Su, Tan K.-L. Indexing high-dimensional data for efficient in-memory similarity search. *IEEE Transactions on Knowledge and Data Engineering*. 2005,Vol. 17, Issue. 3, pp. 339-353

[4] Jaekwan Park, Bonghee Hong, Chaehoon Ban. Efficient Transformation Scheme for Indexing Continuous Queries on RFID Streaming Data. *Second International Conference on Systems and Networks Communications, 2007. ICSNC 2007*, pp.41-47

[5] Jaekwan Park, Bonghee Hong, Chaehoon Ban. A Continuous Query Index for Processing Queries on RFID Data Stream. 13th IEEE International *Conference on Embedded and Real-Time Computing Systems and Applications, 2007. RTCSA 2007*, pp. 138-145

[6] D.V Kalashnikov, S.Prabhakar W.G. Aref and S.E Hambrusch. Efficient evaluation of continuous range queries on moving objects. *Proc.13th International Conference on Database and Expert Systems Applications,2002*, pp.731-740

[7] K. L Wu, S. K Chen and P. S Yu. Processing Continual Range Queries over Moving Objects Using VCR-Based Query Indexes. *International Conference of Mobile and Ubiquitous Systems . 2004*, pp.226-235

[8] Zhenwen He, Chonglong Wu, Cheng Wang. Clustered Sorting R-Tree: An Index for Multi-Dimensional Spatial Objects. *Fourth International Conference on Natural Computation, 2008. ICNC '08.* Vol. 6, pp. 348-352

[9] Proietti G., Faloutsos C. Analysis of range queries and self-spatial join queries on real region datasets stored using an R-tree. *IEEE Transactions on Knowledge and Data Engineering*. 2000,Vol. 12 , Issue. 5, pp. 751-762

[10] Weihua Lin, Yonggang Wu, Xiaojun Tan, Yan Yu. An Improvement of Index Method and Structure Based on R-Tree. *2008 International Conference on Computer Science and Software Engineering* . Vol 4, pp. 607-610

[11] EPCglobal, EPC Tag Data Standards Version 1.3. EPCglobal Standard Specification[S], 2005.

[12] Lifang Yang, Rui Lv, Xianglin Huang, Yueping Liu.

Performance of R-Tree with Slim-Down and Reinsertion Algorithm. *International Conference on Signal Acquisition and Processing, 2010. ICSAP '10*, pp. 291-294

[13] Hung-Yi Lin, Po-Whei Huang. Perfect KDB-Tree: A Compact KDB-Tree Structure for Indexing Multidimensional Data. *Third International Conference on Information Technology and Applications, 2005. ICITA 2005.* Vol. 2, pp. 411-414

[14] EPCglobal, The Application Level Event (ALE) Specification Version 1.1[S], EPCglobal Standard Specification, 2008.

Xiaobo Zhang Lecturer and doctor candidate at the Faculty of Automation, Guangdong University of Technology. His research interests are embedded system, wireless sensor networks, dynamic model and intelligent control.

Lianglun Cheng Prof. and doctoral supervisor of the Faculty of Automation, Guangdong University of Technology. His research interests are network control and integration, information control, embedded system.

Quanmin Zhu Prof. in control systems at the Faculty of Computing, Engineering and Mathematical Sciences(CEMS), University of the West of England (UWE), Bristol, UK. His research interests are nonlinear system modeling, identification, and control.

Surface Water Quality Evaluation Using BP and RBF Neural Network

Qinghua Luan
College of Water Conservancy and Hydropower，Hebei University of Engineering, Handan,056038,China

carol97011202@163.com

Changjun Zhu
College of Urban Construction, Hebei University of Engineering, Handan,056038,China
christorf@126.com

***Abstract*—It is very important to evaluate water quality in environment protection. Water environment is a complicated system, traditional methods cannot meet the demands of water environment protection. In view of the deficiency of the traditional methods, a BP neural network model and a RBF neural network model are proposed to evaluate water quality. The proposed model was applied to evaluate the water quality of 10 sections in Suzhou River. The evaluation result was compared with that of the RBF neural network method and the reported results in Suzhou river. It indicated that the performance of proposed neural network model is practically feasible in the application of water quality assessment and its operation is simple.**

***Index Terms*—water quality;RBF neural network; BP neural network; evaluation**

I. INTRODUCTION

Water quality assessment is an important monitoring project. At present, there are many methods to evaluate the water quality ,such as Fuzzy Comprehensive Evaluation, Unascertained measure, Gray correlation analysis method, gray clustering method, integrated pollution index method[1][2]. Every method has his own characters. These traditional methods can not solve the complicated nonlinear relationship between evaluations indicators and the grade of water quality. Because there are many factors affecting the water quality in river, and the factors is nonlinear relations with water quality. Traditional assessment can not meet the demands for evaluating precision, while artificial Neural networks are loosely based on the neural structure of the brain which provide the ability to learn from the input data they are given and then apply this to unknown data, in effect they can generalize and associate unknown data.

Artificial neural networks(ANNs) are self-organizing ,self-teaching, nonlinear and can deal with the systems which are difficult to described with traditional methods[3][4][5]. In recent years, many researchers have been conducted on the water quality assessment. Liu Lianfang applied BP neural network to the water evaluation of Liao River.Yang Meini allpied Fuzzy artificial neural network to the surface water evaluation in the Shaoguan area. Luo Dinggui designed the RBF model of surface water environment quality assessment [5]. Matlab is mathematical software with high-level numerical computation and data visualization capability. It can provide users with neural network design and evaluation and enable them to work on at greater convenience..

II. CONSTRUCTING THE MODEL WITH ANNS

A. BP neural network model

After the first simple neural network developed by McCulloch and Pitts (1943) , many types of ANN have been proposed. The neural network model with multi-hierarchic structure which is based on back-propagation (BP) arithmetic, the most widely used ANN in hydrologic modeling, is used in this study. The BP network has all the functions of the neural network and its unique advantages such as the good mapping ability of processing the rainfall-runoff partitioning with more flexibility. In addition, its network model contains a highly parallel inter-connection structure, fast self-learning and handling ability of self-adaptation, which provide foundations for the application in hydrology, especially in rainfall-runoff modeling using the incomplete material. Therefore, the BP neural network is a popular choice in the field of hydrology with various complex physical processes.

In recent years, neural network has been widely applied to the teaching evaluation , in which BP network is commonly used . The model created in this paper is a BP neural network with three-layer network (Figure1) .

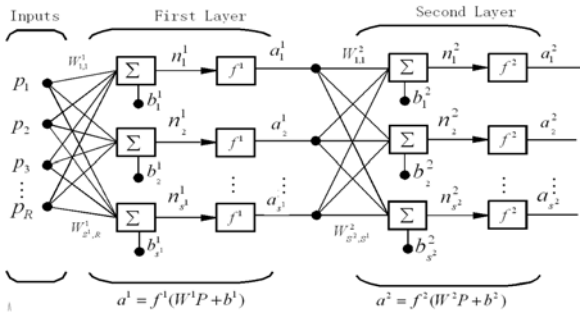


Figure 1. Three-layer BP network structure

doi:10.4304/jsw.6.12.2528-2534

In the Figure 1, p is input of neuron. Each layer has its own weight matrix $\overrightarrow{W}$, its bias vector $\vec{b}$, a net input vector $\vec{n}$, and an output vector $\vec{a}$. $\overrightarrow{W}$ is an $S \times R$ matrix, and $\vec{a}$ and $\vec{b}$ are vectors of length S respectively. The superscripts of symbols identify the layers. Also shown in Figure 1 are R input, S^1 neurons in the first layer, and S^2 neurons in the second layer. Different layers can have different numbers of neurons. The outputs of layers one and two are the inputs for layers two and three. Thus layer 2 can be viewed as a one-layer network with $R = S^1$ inputs, $S = S^2$ neurons, and an $S^1 \times S^2$ weight matrix $\overrightarrow{W}^1$. The input of layer 2 is $\vec{a}^1$, and the output is $\vec{a}^2$. The other layer also can be drawn using same abbreviated notation.

First, the output of the network will be computed. In the hidden and output layers, the net input to unit *I* is of the form:

$$s_i = \sum w_{ji} y_i + \theta_i \tag{1}$$

Several types of transfer functions are used; however, the most frequently used is the sigmoid function. This transfer function is usually a steadily increasing S-shaped curve. The sigmoid function is continuous, differentiable everywhere, and monotonically increasing. In this study, two S-shaped transfer functions in a MATLAB neural network toolbox were used: the tansig function and logsig function. The two functions are of the form:

$$\tan sig(n) = \frac{2}{1+e^{-2n}} - 1 \tag{2}$$

$$\tan sig(n) = \frac{1}{1+e^{-n}} \tag{3}$$

These accumulated inputs are then transformed to the neuron output. This output is generally distributed to various connection pathways to provide inputs to the other neurons; each of these connection pathways.

The process can be described as Figure 2.

B. RBF NEURAL NETWORK

A radial basis function (RBF) is a real-valued function whose value depends only on the distance from the origin, so that $\phi(x) = \phi(\| x \|)$; or alternatively on the distance from some other point *c*, called a *center*, so that $\phi(x, c) = \phi(\| x - c \|)$. Any function φ that satisfies the property $\phi(x) = \phi(\| x \|)$ is a radial function. The norm is usually Euclidean distance.

Radial basis functions are typically used to build up function approximations of the form

$$y(x) = \sum_{i=1}^{N} w_i \phi(\| x - c_i \|) \tag{4}$$

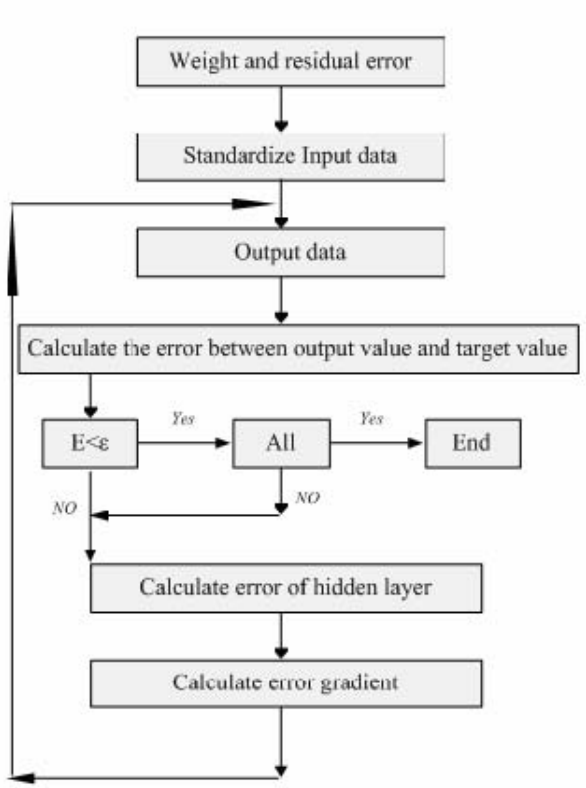


Figure 2. Flowchart of BP neural networks algorithm

where the approximating function $y(\boldsymbol{x})$ is represented as a sum of *N* radial basis functions, each associated with a different center $\boldsymbol{c}_i$, and weighted by an appropriate coefficient w_i. Approximation schemes of this kind have been particularly used in time series prediction and control of nonlinear systems exhibiting sufficiently simple chaotic behaviour, 3D reconstruction in computer graphics (for ex. hierarchical RBF) .

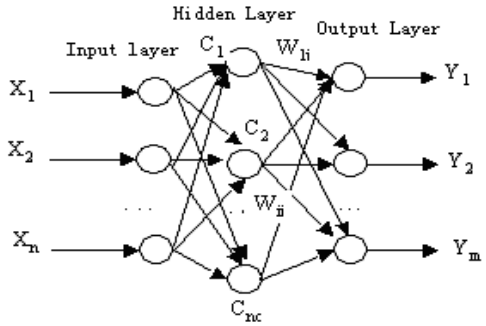


Figure 3. RBF neural network Structure

There are three learning parameters:: center of RBF C_K、σ_K W_K. The process of algorithm as followings:

(1) choosing one series C_K from input matrix;

(2) Calculating variance

$$\sigma = \frac{d_{\max}}{K} \tag{5}$$

Where $d_{\max}$ is the maximum distance, K is the number of C_K;

(3) Calculating $\hat{y}_i(n)$ according to $x(n)$

$$\hat{y}_i(n) = \sum_{k=1}^{M} W_K \phi\left[x(n), C_K, \sigma_K\right] \quad (6)$$

(4) update RBF parameters

$$W(n+1) = W(n) + \mu_w e(n)\phi(n) \quad (4)$$

$$C_K(n+1) = C_K(n) + \mu_C \frac{e(n)W_K(n)}{{\sigma_K}^2(n)} * \phi\left[x(n), C_K(n), \sigma_K\right]\left[x(n) - C_K(n)\right] \quad (7)$$

$$\sigma_K(n+1) = \sigma_K(n) + \mu_\sigma \frac{e(n)W_K(n)}{{\sigma_K}^2(n)} \phi\left[x(n), C_K(n), \sigma_K\right]\left\|x(n) - C_K(n)\right\|^2 \quad (8)$$

$$\phi(n) = \left\{\begin{matrix}\phi\left[x(n), c_1(n), \sigma_1\right], \phi\left[x(n), c_2(n), \sigma_2\right], \\ \ldots, \phi\left[x(n), c_N(n), \sigma_N\right]\end{matrix}\right\}^T \quad (9)$$

$$e(n) = \hat{y}_i(n) - y_d(n) \quad (10)$$

$y_d(n)$ is the expired output; μ_N, μ_C, μ_σ are the echoes of three parameters.

(5) If network convergence, stop the calculation, otherwise go to process (3) . The detail procedures can be seen in Figure.4.

III. APPLICATION

A. Study area

Suzhou is a famous city with much water, in which river port interlocks and numerously lake. water surface in whole city approximately is 3607 square kilometers, which approximately composes the total area 42.52%. The urban district water surface is approximately 24 square kilometers, composing the urban district area 20.15%. Suzhou altogether has more than 4000 bigger lakelet, the big lake swings has 87; Altogether has size rivers 20,000, total length 1457km. Outside the moat the Suzhou old city area circle will become a relatively independent region, spreads across the river course has formed “three horizontal three straight link” the urban river course network of rivers and lakes system with the city. The rivers height is equally 0.8-1.3m (the Woosung elevation) , the water depth is equally 3 meters, the gradient is zero nearly. The river water excretes weakly, therefore the fluent speed of flow is very small, mean velocity of stream is 0.05m/s~0.1m/s.

B. Generation of training set

Water quality is divided into five grades according to the Surface Water Environmental Quality Standard (GB3838-2002) issued by the government of China [7]. Table I shows the Surface Water Environmental Quality Standard BP neural networks have been trained with grade of standard evaluation indicator matrix as input and the grade of category matrix as target outputs[8][9].

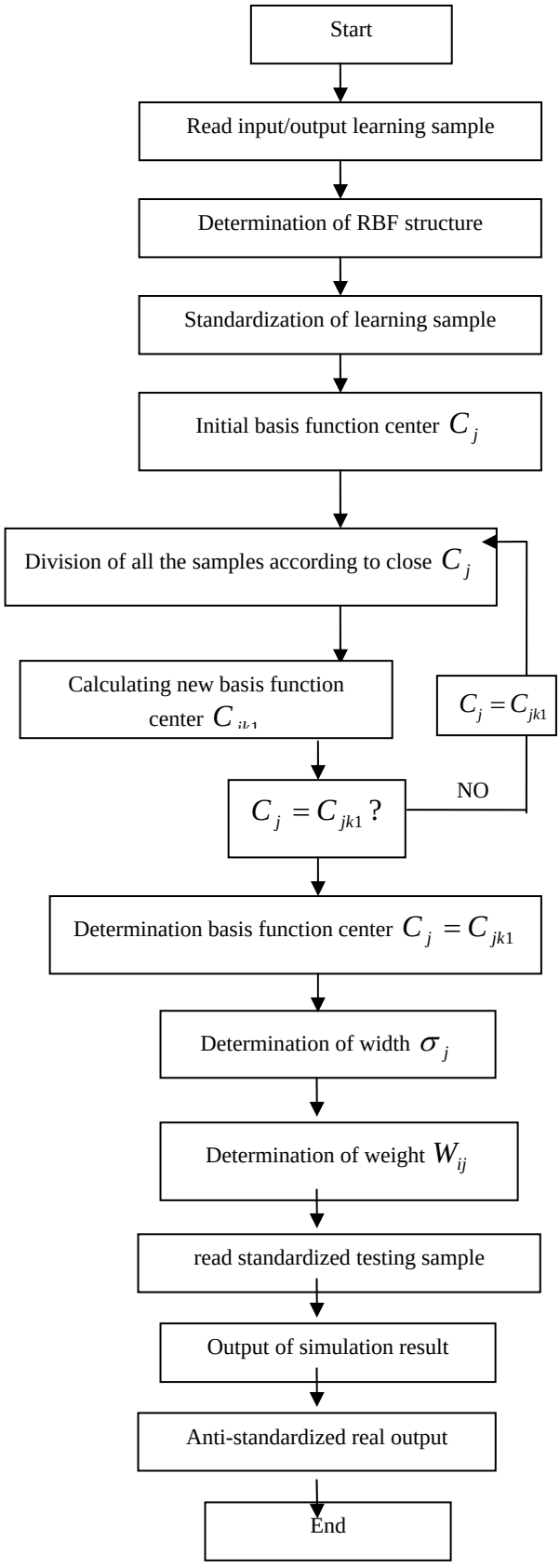


Figure 4. Flow chart of RBF neural network

TABLE I. SURFACE WATER ENVIRONMENTAL QUALITY STANDARD

Indicator j	Ⅰ	Ⅱ	Ⅲ	Ⅳ	Ⅴ
DO	>7.5	6.0	5.0	3.0	<2.0
BOD_5	<3.0	3.0	4.0	6.0	>10.0
COD_{cr}	<15.0	15.0	20.0	30.0	40.0
NH_3-N	<0.015	0.5	1.0	1.5	2.0
Phenol	0.002	0.002	0.005	0.01	0.1

The standard target outputs of the five grades are (1,0,0,0,0) ,(0,1,0,0,0) ,(0,0,1,0,0) ,(0,0,0,1,0) ,(0,0,0,0,1) . The result of target outputs can be seen in Table II.

C. Pre-processing of data

Data pre-processing is an important initial work. The process includes training sample's size, criterion unitizing transforms (unification replacement) , the statistical property research, spatial information processing as well as singular value processing and so on. Before the data is processed, the data should bedivided into two groups: training sample and examining sample. There are several types of preprocessing methods. All the training data are rescaled to a specific range (e.g., [-1, 1] or [0, 1]) . However, the BP model is based on the gradient descent algorithm and the transfer function which determines the relationship between inputs and outputs of a node and a network has an asymptotic nature. When the extreme values of discharge are utilized, the gradient of the transfer function will approximate to zero, consequently, leading to slow the training down.

The next issue is the division of the data into the training and testing data set. The training data set is presented repeatedly to the network until the weight values are determined while the testing data set is used for the final evaluation of the BP model performance. Sometimes to overcome the problem of overfiting or to determine the stopping point of the training process, the validation set is also required. The testing data set is used for both validation and testing in this study. Although there are no general solutions to the selection of the training and testing set which may affect the performance of the BP model, both the training and testing sets representative of the evaluation system data should be carefully evaluated in the decision making

D. Water quality evaluation model

1) BP evaluation Model

(1) Neuron determination of input layer

According to the indicator system, the main index affecting teaching quality is 5 (Table I), so the number of input layer can be adopted as 5. The determination of input pattern is the key for the successful neural network model. If the input neurons are more, the structure of model is more complex and training period is long. Otherwise, it is difficult to get the nonlinear relationship.

(2) The number of hidden layers and nodes

ANNs perform complicated nonlinear mapping between input and output variables through the hidden nodes in the hidden layer. It can capture the pattern in the data. So the hidden layer and nodes play very important roles in the network architecture. Most studies indicate that a single hidden layer network tends to be used to the modeling problem and if the number of hidden nodes is enough, we can obtain any desired accuracy. However, other studies demonstrated the benefits of an ANN comprised of two hidden layers . In the recognizance of these studies, a single hidden layer is used in this study.

The number of hidden nodes has comparatively great difficulty in determining. The neuron selection in the hidden layer affects the precise calculation and learning efficiency for the whole BP network, up till now, there are still no unified ways to identify the number of the hidden layer neuron, which is at the stage of research and exploration. If a small number of the hidden layer neurons are chosen, the self-adaptability of the BP neural network will be reduced, thus the training results are not ideal. However, if the large number is chosen, it is generally believed that the time for network training will be greatly increased, and some meaningless information in training data will be remembered, thus it will reduce precise calculation in a sense. The most common way in determining the number of hidden nodes is by using experiments or by trial-and-error. Thus in this study the more appropriate number of the hidden layer neuron is fixed for six through testing and comparing. Therefore, the chosen configuration for the BP model is 5-8-5: five inputs, eight hidden neurons in one hidden layer and five outputs.

TABLE II. PERFORMANCE COMPARISON IN DIFFERENT TOPOLOGY OF BP NEURAL NETWORK

BP Neural Network	Topology	Epochs	mse
BPⅠ	5-7-5	1871	0.00000997887
BPⅡ	5-8-5	1434	0.00000993658
BPⅢ	5-9-5	2128	0.00000998447
BPⅣ	5-10-5	2066	0.00000994926
BPⅤ	5-11-5	1652	0.0000099837

(3) Results

According to the BP network test (simulation identification), the water environmental quality on the 20 monitoring sections in Suzhou River are evaluated. The results are shown in Table II.

Ⅱ) RBF neural network

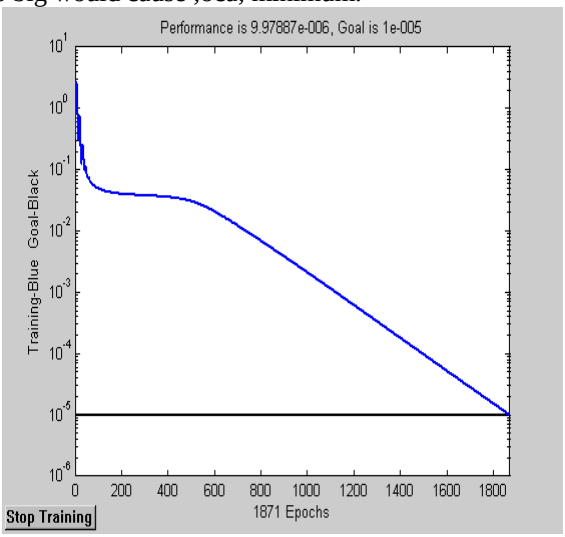
(1) Creation of Rbf neural network

RBF network input layer neurons depends on the number of indicators of water quality evaluation. According to meaning of the questions identified as 5, the output layer neurons is set to 5, the use of MATLAB 6. 5 in the NEWRB functions to create the network, automatically determines the required hidden layer unit number.Hidden layer unit activation function for the RADBAS

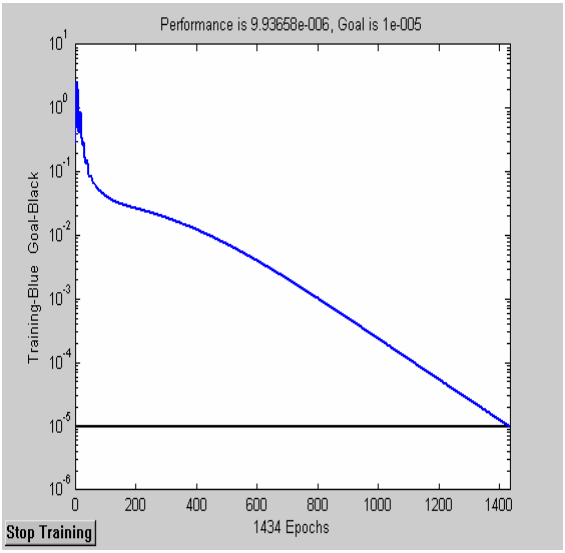
(2) Training, testing and evaluation

The process of creating RBF neural netwok is also a learning process. The important parameter is SPREAD. The parameter is the pace of expansion of radial basis function. Its value can not be too small and not too big,

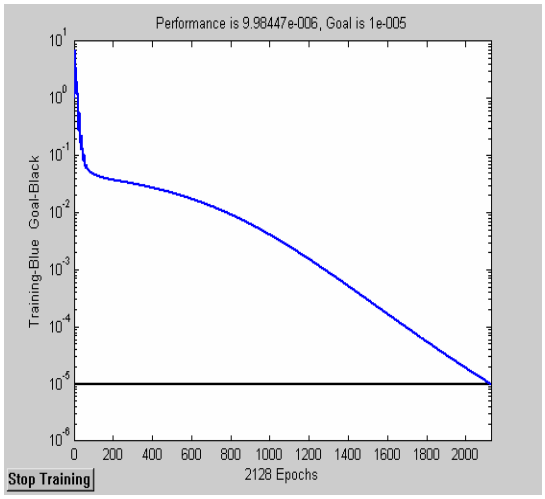
too small will affect the network's convergence speed, too big would cause ;oca; minimum.



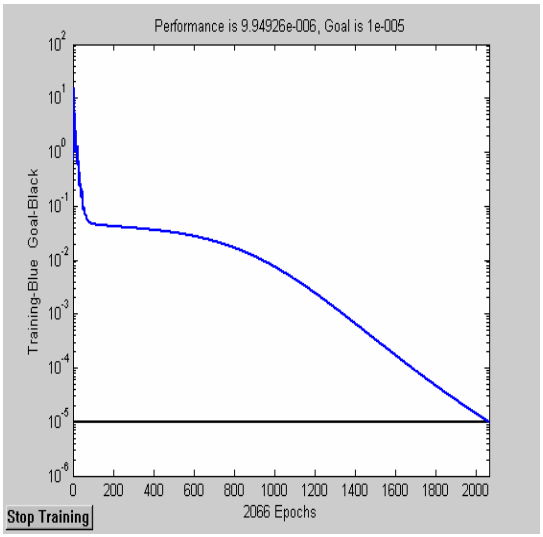
(a) 5-7-5



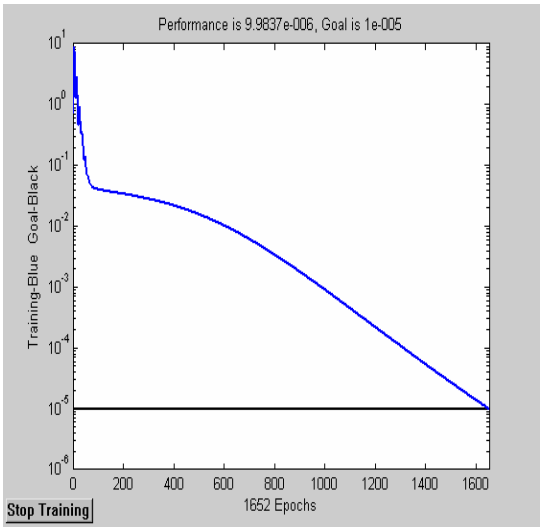
(b) 5-8-5



(c) 5-9-5



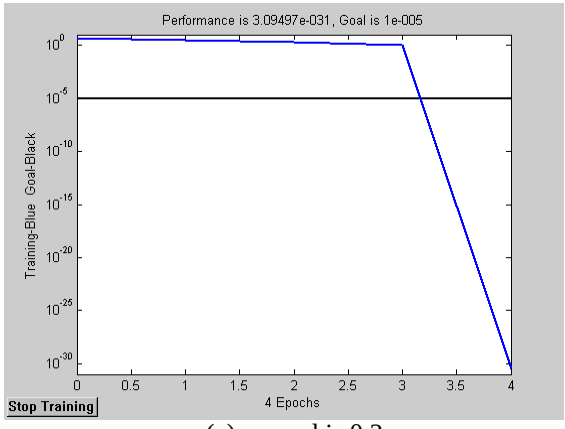
(d) 5-10-5



(e) 5-11-5

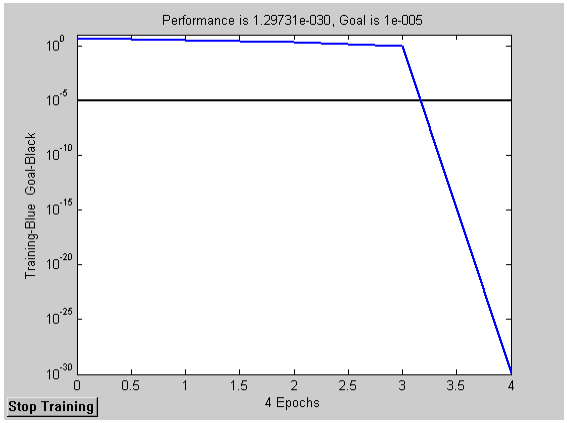
Figure 5. Relation of different topology with epochs

In the process, different SPREAD value is needed to detemine an optimal value. In this study, the value of SPREAD is 0.2, 0.25, 0.4, 0.45, 0.55,which is called RBF I, RBF II, RBF III, RBF IV, RBF V.



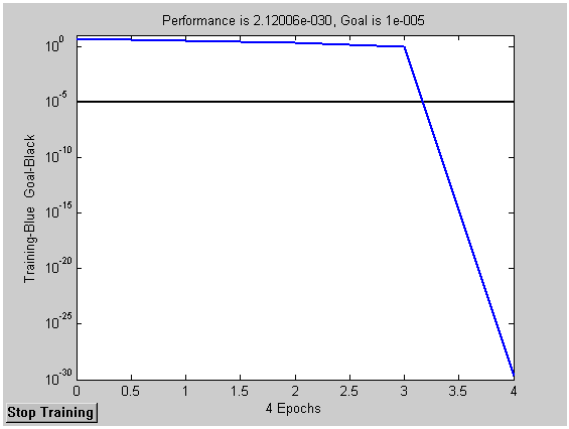
(a) spread is 0.2

After 4 epochs, network is convergence, the final square error E(g1) =mse $_{I}$ =3.09497e-031



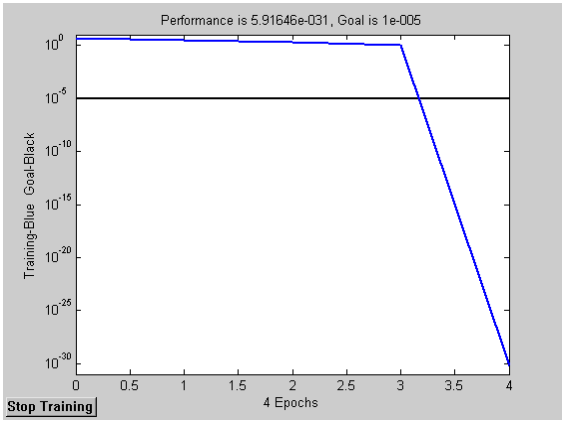
(b) spread is 0.25

After 4 epochs, network is convergence, the final square error E(g2) =mse $_{II}$ =1.29731e-030



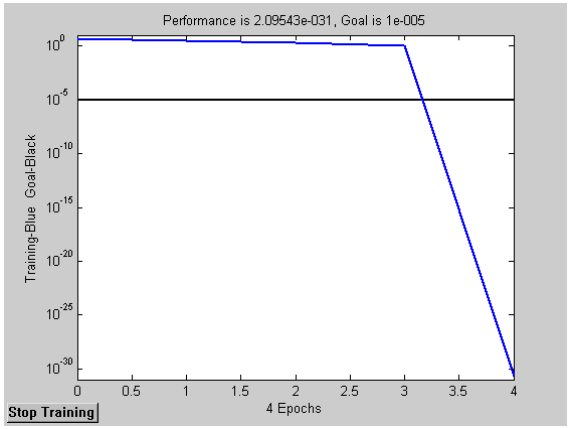
(c) spread is 0.4

After 4 epochs, network is convergence, the final square error E(g3) =mse$_{III}$=2.12006e-030



(d) spread is 0.45

After 4 epochs, network is convergence, the final square error E(g4) =mse$_{IV}$=5.91646e-031



(e) spread is 0.55

After 4 epochs, network is convergence, the final square error E(g5) =mse$_{V}$=2.09543e-031

Figure 6. Relation of different SPPREAD with epochs

TABLE III. PERFORMANCE COMPARISON IN DIFFERENT TOPOLOGY OF RBF NEURAL NETWORK

RBF Neural Network	Topology	Epochs	mse
RBF I	0.2	4	3.09497e-031
RBF II	0.25	4	1.29731e-030
RBF III	0.4	4	2.12006e-030
RBF IV	0.45	4	5.91646e-031
RBF V	0.55	4	2.09543e-031

Seen from Table III and IV, compared to other models, RBF V optimal network performance model, the model in the loop 4 times after the network convergence, the final mean square error of 2.09543e-031, for Table III, models in the minimum mean square error of the model, so this Suzhou Creek water quality evaluation should use RBF V Network

IV. CONCLUSIONS

This paper adopts the BP and RBF neural network to evaluate the water quality in Suzhou river. Some conclusions can be got by our research:

(1) BP and RBF neural network can be used to evaluate the water quality. The results are objective.

(2) The result of evaluation through BP neural network has high precision. RBF's convergence is fast than BP's

(3) The assessment result can provide reference to the water environment protection and plan kind of pagination.

ACKNOWLEDGMENTS

This study was funded by the National Basic Research Program of China (2010CB951101) , the National Natural Science Foundation of China (Grant No. 40830639, 50879016, 50979022 and 50679018) , the program for Changjiang Scholars and Innovative

Research Team in University (IRT0717) and the Special Fund of State Key Laboratory of Hydrology-Water Resources and Hydraulic Engineering (1069-50985512, 1069-50986312)

REFERENCES

[1] Chen T,Chen H. Approximation capability to functions of several variables, nonlinear functions and operator by radial basis function neural network[J].IEEE Trans on neural networks,1995,(6)

[2] Martin T. Hagan, Howard B. Demuth, Mark H. Beale, Neural network design, China Machine Press, Beijing, 2004.

[3] L.M. Zhao, H.Y. Hu, D.H. Wei, S.Q. Wang, Multilayer forward artificial neural network, Yellow River Conservancy Press, Zhengzhou, 1999.

[4] Fu Yongfeng, Zhang jian , Luo Guangming. Application of BP network to groundwater quality evaluation[J].Journal of Northwest Sci-Tech University Agri. And Fore., 2004,32(11) :129-132.

[5] Luo Dinggui, Guo Qing, Wang Xuejun. Neural network model design of surface water environmental quality assessment[J]. Geography and Geo-Information Science, 2003,19(5) : 77-81 (in Chinese)

[6] Zhu Changjun, Li wenyao, Zhang pu. Application of Artificial Neural network in water environment quality assessment[J].Industrial Safety and Environmental Protection,2005,31(2) :27-29.

[7] X. Yuan, H. Li, S. Liu, and G. Cui, Nerve network and heredity algorithm in water scientific domain application, Water Power Press Beijing, China, pp. 15-16. 2002.

[8] Feisi Research and Development Center of Science and Technology, MATAB 6.5 auxiliary neural network analysis and design, Electronics industry press, Beijing, 2003.

[9] X.P. Cao, C.H. Hu, "Fault prediction for inertial device based on LMBP neural network", Electronics Optic and Control, 2005, 12(6) , pp. 38-41

[10] Cheng Wanli, Li Yifang, Hao Fuqin. The Evaluation of Water Quality of Sanmenxia Reach Based on Fuzzy Math [J].Environmental Science and management, 2007,32(10) :188-190

TABLE IV. EVALUATION RESULTS OF BP NEURAL NETWORK IN SUZHOU RIVER

Section	BP	RBF	Function zone
1	IV	IV	IV
2	IV	IV	IV
3	IV	IV	IV
4	V	IV	IV
5	IV	IV	IV
6	I	V	V
7	I	IV	V
8	IV	V	V
9	IV	IV	IV
10	IV	IV	IV

Call for Papers and Special Issues

Aims and Scope.

Journal of Software (JSW, ISSN 1796-217X) is a scholarly peer-reviewed international scientific journal focusing on theories, methods, and applications in software. It provide a high profile, leading edge forum for academic researchers, industrial professionals, engineers, consultants, managers, educators and policy makers working in the field to contribute and disseminate innovative new work on software.

We are interested in well-defined theoretical results and empirical studies that have potential impact on the construction, analysis, or management of software. The scope of this Journal ranges from the mechanisms through the development of principles to the application of those principles to specific environments. JSW invites original, previously unpublished, research, survey and tutorial papers, plus case studies and short research notes, on both applied and theoretical aspects of software. Topics of interest include, but are not restricted to:

- Software Requirements Engineering, Architectures and Design, Development and Maintenance, Project Management,
- Software Testing, Diagnosis, and Validation, Software Analysis, Assessment, and Evaluation, Theory and Formal Methods
- Design and Analysis of Algorithms, Human-Computer Interaction, Software Processes and Workflows
- Reverse Engineering and Software Maintenance, Aspect-Orientation and Feature Interaction, Object-Oriented Technology
- Component-Based Software Engineering, Computer-Supported Cooperative Work, Agent-Based Software Systems, Middleware Techniques
- AI and Knowledge Based Software Engineering, Empirical Software Engineering and Metrics
- Software Security, Safety and Reliability, Distribution and Parallelism, Databases
- Software Economics, Policy and Ethics, Tools and Development Environments, Programming Languages and Software Engineering
- Mobile and Ubiquitous Computing, Embedded and Real-time Software, Database, Data Mining, and Data Warehousing
- Internet and Information Systems Development, Web-Based Tools, Systems, and Environments, State-Of-The-Art Survey

Special Issue Guidelines

Special issues feature specifically aimed and targeted topics of interest contributed by authors responding to a particular Call for Papers or by invitation, edited by guest editor(s). We encourage you to submit proposals for creating special issues in areas that are of interest to the Journal. Preference will be given to proposals that cover some unique aspect of the technology and ones that include subjects that are timely and useful to the readers of the Journal. A Special Issue is typically made of 10 to 15 papers, with each paper 8 to 12 pages of length.

The following information should be included as part of the proposal:

- Proposed title for the Special Issue
- Description of the topic area to be focused upon and justification
- Review process for the selection and rejection of papers.
- Name, contact, position, affiliation, and biography of the Guest Editor(s)
- List of potential reviewers
- Potential authors to the issue
- Tentative time-table for the call for papers and reviews

If a proposal is accepted, the guest editor will be responsible for:

- Preparing the “Call for Papers” to be included on the Journal’s Web site.
- Distribution of the Call for Papers broadly to various mailing lists and sites.
- Getting submissions, arranging review process, making decisions, and carrying out all correspondence with the authors. Authors should be informed the Instructions for Authors.
- Providing us the completed and approved final versions of the papers formatted in the Journal’s style, together with all authors’ contact information.
- Writing a one- or two-page introductory editorial to be published in the Special Issue.

Special Issue for a Conference/Workshop

A special issue for a Conference/Workshop is usually released in association with the committee members of the Conference/Workshop like general chairs and/or program chairs who are appointed as the Guest Editors of the Special Issue. Special Issue for a Conference/Workshop is typically made of 10 to 15 papers, with each paper 8 to 12 pages of length.

Guest Editors are involved in the following steps in guest-editing a Special Issue based on a Conference/Workshop:

- Selecting a Title for the Special Issue, e.g. “Special Issue: Selected Best Papers of XYZ Conference”.
- Sending us a formal “Letter of Intent” for the Special Issue.
- Creating a “Call for Papers” for the Special Issue, posting it on the conference web site, and publicizing it to the conference attendees. Information about the Journal and Academy Publisher can be included in the Call for Papers.
- Establishing criteria for paper selection/rejections. The papers can be nominated based on multiple criteria, e.g. rank in review process plus the evaluation from the Session Chairs and the feedback from the Conference attendees.
- Selecting and inviting submissions, arranging review process, making decisions, and carrying out all correspondence with the authors. Authors should be informed the Author Instructions. Usually, the Proceedings manuscripts should be expanded and enhanced.
- Providing us the completed and approved final versions of the papers formatted in the Journal’s style, together with all authors’ contact information.
- Writing a one- or two-page introductory editorial to be published in the Special Issue.

More information is available on the web site at http://www.academypublisher.com/jsw/.

(Contents Continued from Back Cover)

REGULAR PAPERS